Biometric Systems

INTRODUCTION

The term biometrics is derived from the Greek words bios (life) and métron (measure). Biometrics refers to the study and use of methods for detect and measure the characteristics of living organisms and draw comparatively classifications and laws. Finds applications in biology, medicine, genetics, in the agricultural and forestry sciences, environmental science and other related fields.
The modern meaning of the term biometrics used in Computer Science, and consequently of the term biometric system, explicitly mainly refers to the automatic identification or verification of the identity of a person based on physical or behavioral characteristics.
Biometric Systems are just special systems that carry out a particular kind of pattern recognition.
The **Pattern Recognition** is something that is somehow at the basis of many technological fields like Computer Vision, Natural Language Processes. Speech Recognition, etc.
In practice, a pattern is a special model that somehow brings informations that is necessary to summary the characteristics of a certain object. Such characteristics are usually called features. So we have features that have been extracted in order to create a model. Once we have extracted these features, we must alto device a way to measure the distance between two different groups of features in order to establish if they come from the same object or class of objects or not.
Two patterns are similar if the measure of the distance between their feature vectors is small (**three basic issues**: what is a good distance measure, which are the best features, what is the difference margin to accept).
What is a good distance measure? It depends of the kind of features. First we decide the features that we want to extract.
What is a good feature? What is a good group of features? They are able to distinguish either different classes or two different individuals in the same class. So a feature or a set of features is good if it is discriminative enough for our kind of task.
The **distance measure** is easy to compute, quick to compute and reliable in the sense that actually reflects the difference between feature vectors. We must be flexible enough to be able to recognize two models even though they are not exactly the same, but at the same time we must also taking into account what can happen in this process. Because this process can also poison the recognition.
We can carry out pattern recognition in different ways. This is specially evident in content based image retrieval. For example, we can have that classes are types of objects. What is a pattern in this case? When my class is a type of objects, a pattern is any kind of model or set of features that allows to distinguish for example a face from a flower.
The level of details that are required is different because of course if we want to distinguish people from panorama we can use less detailed information. But we can go a more detailed level of classification. For example, if we have our classes that became subclasses of the same type (for examples the classes are dogs and the subclasses are different kinds of dogs). So at this point, even the pattern must became more detailed and more suited for this higher level of details. Because at this point we don't only want to distinguish that we have a dog in the picture, but we also want to know which is the specific kind of dog.

When we go to biometrics, we have the higher level of details in general because in the classical biometrics problems (verification and identification) our classes are the single individual. Each individual is a single class.
So at that point we want to build a model able to distinguish each and any individual that is relevant for the system.
So in this case the pattern allows to distinguish among individuals.

What is biometrics?

In general biometrics measure and statistical analysis of biological data. In technological sense biometrics measure and analysis of physical and/or behavioral characteristics to authenticate /recognize a person.
The Biometric Consortium define biometrics as “automatic recognition of a person according to discriminative characteristics”.
The **basic assumption** is that each person is unique. Is this true? It depends on the characteristics that we consider.
The **main issues** are: determine the unique features able to identify a person; find reliable techniques to measure such features; devise reliable algorithms to recognize/classify a person according to the measured features.

ACCESS TYPES

The most used application for biometrics is access type. "Access Types" mean authentication or authorize people so that they can the access to either the physical or logical resources only if they have been granted such access.
The most typical kind of access is the **Physical Access** (room, building, in a determinate area). For example is possible to set up an authentication in order to enter in a room. This authentication can be carried out using fingerprints or even using face recognition.
We can also mention the **Logical Access**, that are for example a remote access to electronic resources or to critical data.

WHY BIOMETRIC SYSTEMS

At present, recognition (often for authentication purposes) is performed according to two modalities:

- **Something one owns**: a card or a document, but it can be lost, or stolen, or copied. Actually the system authenticates the object, not its owner.
- **Something one knows**: an individual or community password but it can be guessed, wormed out or forgotten.

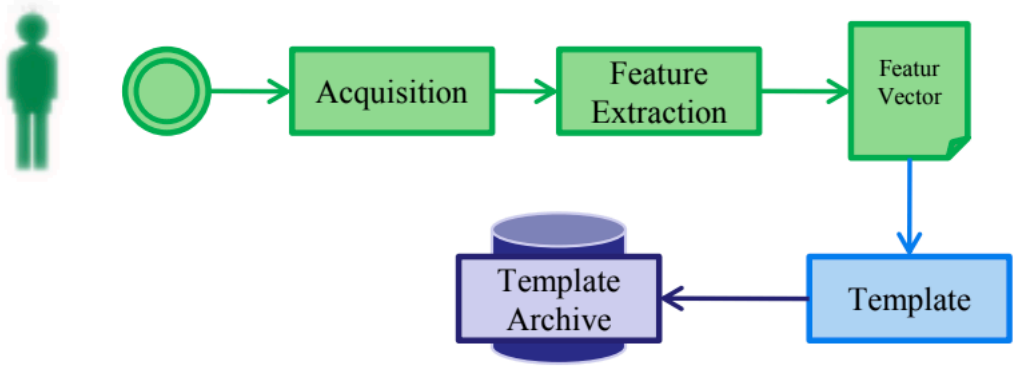
So based upon what one is or, in another way, how a person appears and/or how a person behave. So we can have fingerprints, the voice, palm prints, the iris (the right iris is different from the left iris), the face and signature. That are all characteristics extremely distinctive.
What is the key of biometric system? Is to transform each such feature or even a combination of such features into a digital key. And such digital key should be able to discriminate between different people.
Each biometric trait has different characteristics, so they can be coded using different features.

BIOMETRICS DOES NOT ALWAYS WORK

Biometrics is not the solution for all problems because, for example, a pattern matching that is applied to a natural phenomena, it may encounter problems when we have very similar persons or we have some kind of distortion that can cause people to seem similar. But the integration with traditional systems is the real key for a better use of biometrics.

ARCHITECTURE OF A BIOMETRIC SYSTEM

Enrollment: Capture and processing of user biometric data for use by system in subsequent authentication operations (gallery). We cannot recognize something that has not been registered before. So we can only recognize people if we have enrolled such people in our system. The Enrollment is the process of capturing and processing user biometric data in order to be used in the following authentication.



We have "**acquisition**" first, that means to capture in some way the biometric traits. The way we capture the traits has to be suited to the trait itself (for example, the face is usually captured by a photo, but if we want to process a 3D model of the face, we should use range cameras).

Once we have the sample (what we've captured), we have to extract feature, so we the process of "**feature extraction**". What are **features**? The minutiae in fingerprints and their position; the geometric relationship in a face or the texture of the face; and so on and so forth.

With these features we build the so called "**feature vector**". These last two procedures are not necessarily strictly bound each other. If for example we have a huge amount of features, we want to process them all or store them all, because more of them can be useless or not distinctive enough. So in general there are processes that try to reduce the dimension of the features space. Once this process has been carried out, we pass from feature extraction to build the feature vector.

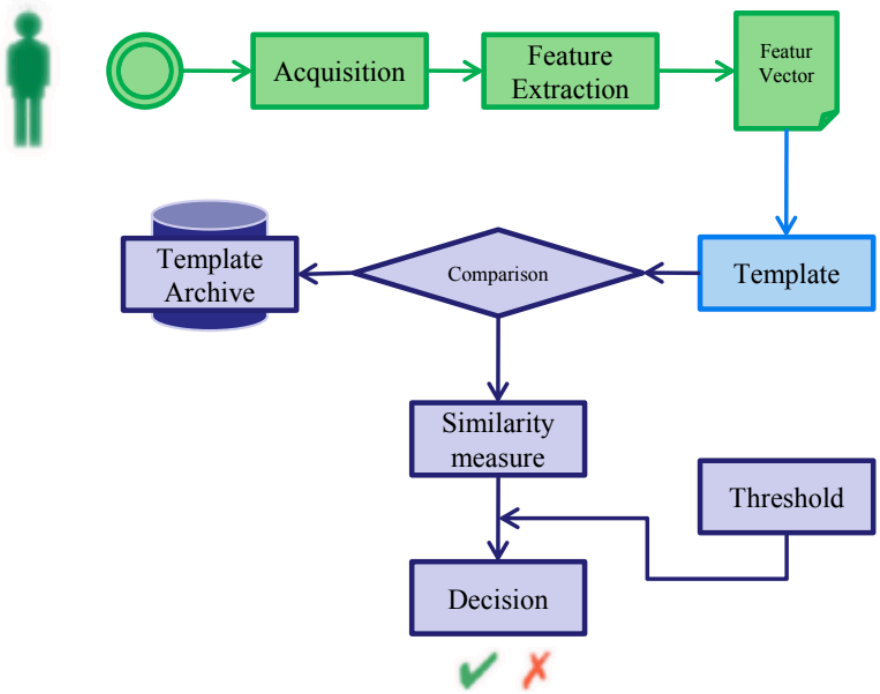
Such feature vectors together with other possible informations makes up the "**biometric template**". The template enters in the so called gallery of the system ("**template archive**").

What is the **gallery** of a biometric system? The gallery is the set of templates that appertaining to an enrolled subject. We can have more templates for the same subject in order to make the the process of recognition more accurate, because, for example, we can recognize this person with different poses.

Recognition: Capture and processing of user biometric data in order to render an authentication decision based on the outcome of a matching process of the stored to current template (verification 1:1 identification 1:N).

What does recognition entail? Recognition entails that, we have for a certain point the exact processes that we had for the enrollment (so we have acquisition for which we should use the same kind of sensor in general, we have the same feature extraction and feature vector construction, we build the template) but after build the template, this time instead of store it, it will be compared against to the gallery.

The sample in input has a special name that is called probe. Probe is defined as the template coming from the input sample that we have to submit to the system for recognition.



So the comparison between the template and the templates in the archive provides a similarity measure (such similarity measure can be computed either for a single identity or for multiple identities) that must be effective enough to distinguish. According to this, in most cases, we apply the acceptance threshold after which we can take the decision: whether to accept the recognition or not.

We use an **acceptance threshold** that has different enroll according to whether we're using a **similarity measure** or a **distance measure**. We can use both at the same way. If we use the similarity measure, the threshold is a lower limit, that means if we have the similarity lower than the threshold then we reject (the recognition is negative); if we have the similarity value above the threshold, then we accept (the recognition is positive).

If we using the distance instead the threshold becomes an upper limit. We can accept the recognition as positive only when the distance falls below the threshold; if the distance is above the threshold, then we have to reject the recognition. So we have the opposite of the similarity.

Probe: each template which is submitted for recognition.

Gallery: the set of templates pertaining to enrolled subjects.

We may have either verification of an identity or identification. What is the difference?

Verification means the person claims an identity which is the typical situation of authentication. For example, i clam an identity and I submit the my sample as a probe; the system has one or more of my templates in the archive; matches this or those templates with the incoming template; before decide if that person is who he claimed to be, we compare the similarity compared by the matching with the acceptance threshold (this threshold is decided in advance, when designing the system); if the similarity measure meets the acceptance threshold, then I accept the claim, otherwise we reject the claim as an impostor. So from the point of view of the identities is a one-to-one matches: we only match the incoming template with the single or the multiple templates in the gallery that belong to the claimed identity. On the other hand, in the **identification** there is no identity claim, like in a video surveillance: we don't know who are the persons in the room but we want identify each person. We take each probe and we compare them with the overall gallery. So from the point of view of the identity, we have a one-to-N matching,

because we have a single probe in input but we have to compare it with all the probe set that we have in the gallery unless we apply something to cut the search.
Identification may not use the threshold in order to decide if the most similar template that has been identified in the archive can be actually attributed to that probe or not.

MODULES OF A BIOMETRIC SYSTEM

According to the architecture that we assume, we can also identify different modules that are more or less independent from each other. We said "more or less" because they have some kind of dependences.
A biometric system is generally designed to operate with four modules.

1. **Sensor Module** : where biometric data are caught.
2. **Feature extraction module** : where a set of main characteristics is extracted from acquired data. During enrollment it produces the templates to be stored in the system. We cannot extract features which are not compatible with the kind of sample that has been captured. So there is a implicit dependency between the kind of sample that we have extracted and the kind of features that we can extract from such sample. This "Feature extraction module" provides the data for building the feature vectors involved in recognition.
3. **Matching module**: where extracted features are matched with stored templates to return one or more matching scores. It does the matches with the features that have been extracted. This "Matching module" provides a result that can be a similarity value or can be a distance value and then, according to this, the decision module uses a suitable threshold can provide the farther response to the user.
4. **Decision module**: where a decision is made according to matching results.

In general, a recognition operation entails matching to different templates: templates with homogeneous features extracted from samples of the same type. However we can also have a different kind of model that it is, for example, entailed in Deep Networks and Machine Learning approaches, where in practice we don't have a 1 to 1 comparison template against template, but rather we build a model of the class of we want to recognize and, when a new probe arrives, the template extracted is submitted to the model to see if it's consistent with that model. So that is not a real matching of feature vectors, but it's a kind of a classification process that it takes the template in input and decides if this template is consistent with the model produced during a training phase.

How can we measure the performance of a Biometric System?

According to the kind of recognition application that we have, we will apply different strategies in order to measure the system performances. Because according to the kind of application, the System can do different kind of errors and our performance evaluation will be most based on errors (measures the number of errors that the system does with respect to the correct responses.
An other important thing to taking into account when we design a biometric application is the **type of users** that could be involved in our application.
Most of the users will be **cooperative**: the user is interested in recognition (an impostor might try to be recognized as a legal user). On the other hand, we have the **non-cooperative**: the user is indifferent or even adverse to recognition (an impostor might try to avoid being recognized). **Public/Private**: users of the system are customers or employees of the entity installing the system. The users of the system can be customers of a certain organization and in this sense they can be more or less cooperative. **Used/Non used**: frequency of use of the biometric system (more times a day, daily, weekly, monthly, occasionally ...). **Aware/Not aware**: the user is aware or not of the recognition process.

The two most important classifications are cooperative/non-cooperative and the two most important types of settings are controlled/uncontrolled.
In the **Controlled setting** the capture conditions are well designed in order to get the better result. The capture settings can be controlled, the distortions are mostly avoided (e.g., for face, pose, illumination, and expression), defective templates can be rejected, and capture repeated. In the **Uncontrolled/Undercontrolled** case, the capture settings cannot be controlled, template can present various levels of distortion, defective templates can be rejected, but capture cannot be repeated.
It's important to consider whether the user is cooperative/aware of recognition (this means that he will do his best to be recognize, so we can relax some constraints of the captured samples) or not-cooperative/not-aware (line in the case of criminals that don't wont to be recognized).
The other kind of classification of applications is related to the environments. The setting of the acquisition of the system can be either controlled (we can get some good features of the captured samples) or uncontrolled.

TYPES OF RECOGNITION OPERATION

Face recognition is generally divided into two sub-categories. On the one side, **face verification** (or 1:1 face recognition) consists in checking if a face corresponds to a given identity. On the other side, **face identification** (or 1:N face recognition) consists in finding the identity corresponding to a given face.
In **Verification** we have a user claim. The user claims an identity, possibly by presenting an ID card or other additional stuff. The system performs a 1:1 matching to verify the claimed identity. If the system verifies that the person is the same, the user is accepted. If the response is negative, the user doesn't correspond to the the enrolled user. In recognition operation of verification type, we have different samples of the same individual (one could be the claimed identity, the other one could be the enrolled identity).
In **Identification** we don't have claim by the user. But we have an incoming probe and the system has to determine the correspondence with one of the subjects in the system gallery by a 1:N matching operation. What we get from the system in this case is a possibly list of candidates. In general the first recognized identity is returned. In this case the kind of evaluation is this: the identity is correct or not. Is much more interesting to analyze the possible longer list of candidates in order to see where the correct identity was apart.
Identification does not entail any kind of identity claim; the system try to recognize who is the person that is trying to enter in the system. There are two types of Identification:

- **Open set**: the system determines if probe belongs to a subject in the gallery G. It may happen that some subject doesn't belong to the gallery. While in the closed set the only possible response is an identity that can be the correct one or not according to the accuracy of the system, in open set identification we must add a reject option, because the system must also recognize the case when the incoming subject is not included in the system gallery.
  In this case, the **possible error** is reject a probe belonging to an enrolled subject.
- **Closed set**: the possible error in this case is the we can return the wrong identity, but the same can happen in identification open set, because besides rejecting an identity that is actually enrolled, we can accept the identity; but we can return a wrong person.
- **Watch list**: the system has a list of subjects and checks if the probe belongs to the list. **White list**: subjects in the list are granted access; **Black list**: subjects in the list are rejected (possible alarm).

REQUIREMENTS FOR A BIOMETRIC TRAIT

Characteristics that a trait must have in order to be considered a biometric trait. Biometric trait is whatever pertains the appearance of a person or his behavior. Some traits are **harder** than others: in order to be an **hard biometric trait**, that can be used whatever is our setting or with whatever kind of user we have, this trait must be universal. An **universal biometric trait** must be owned by any person (except for rare exceptions). So, all people that possibly enter in the system must posses this trait. The biometric trait must be also unique. The **uniqueness** of a biometric trait means that any pair of people should be different according to the biometric trait. Uniqueness ensures that we can distinguish people to persons according to that trait.
Another important requirement for a biometric trait is **permanence**: the trait must be permanent, so it mustn't change in time. **Collectability** is important because we cannot use the biometric trait especially in a large scale if it is hardly collectable. So the biometric trait should be measurable by some sensor.
Lastly there is **acceptability**: involved people should not have any objection to allowing collection/measurement of the trait.
We can have also a **soft biometric trait**: it is a trait that can be used either to cut the search or to improve the accuracy of the system; it can be useful but lacks some of such features. They are obtained relaxing the uniqueness; we can obtain some biometrics traits that while not being useful in order to recognize the single person, can be useful in order to recognize a group of person in order to reduce the search.

ACKNOWLEDGED TECHNIQUES IN X9.84 - 2003 STANDARD

We have **world wide standard** in order to ensure a minimum of security for biometric traits. We have ISO standards that describe which are the better settings to capture biometric traits and also the characteristics that a good sample must present.
For example, for the fingerprints biometry we have the most huge quantity of standards regarding the quality of both the capture devices and the capture samples pertains to fingerprints. This because fingerprints are frequently used also in forensics so not only for authentication.
Regarding eye biometry we have standards related both to iris and retina recognition.
Even for the face biometry we can use different capture technologies, for example we can have photo in visible light and we can also have a photo in infrared (night images).
Taking into account also the ear biometry and the hand biometry (that is based on finger geometry), these 5 types of biometrics are based on physiological features.
Another class of biometric features is represented by **behavioural features**: signature biometry and keys typing.
Regarding the voice biometry we've in the mixed features. Voice recognition is a kind of mixed features because there are physiological elements that produce the voice (the anatomical parts of body that are involved in voice production) and behavioural elements somehow influence the voice (the state of mind of a person).
Lastly we have biological traces like DNA.

BIOMETRICS

- **Strong Biometric Traits**: Iris; Face (changes slowly during a certain range age); Fingerprints.
- **Soft Biometric Traits**: Hair Color; Facial Shapes (can be used to cut the search; but there is a problem because if a person loses or gains weight, the face shape changes); Gait (can be affected by the kind of ills, by the kind of shoes, by possible different ground conditions).

The Soft Biometric Traits either lack uniqueness (e.g., hair color) or persistence (e.g., behavioural that are affected by mood, health, etc.) but can be used to limit the search.
Given that samples have different natures, also the features that we extract and the techniques that we used must be suitable for the form of the samples.
For example, the **Voice** is made up from some waves; so one way to create a feature or a model for the voice is to exploit a Gaussian Mixture Model (GMM). This is a statistical model that uses a mixture of gaussians in order to reproduce the original signal. There are also special acoustic vectors that are used for training because in general for voice recognition we train a system to recognize a voice according to these kind of features.
In **speaker recognition** we would like to recognize the person who is speaking: we may have either text dependencies to speaker recognition systems so that we're able to recognize the person only when he pronounces a conventional pass phrase or text independencies to speaker recognition that means to recognize the person whatever this person pronounces.
On the other hand, in **speech recognition** we've a symmetric distinction: in the speaker recognition the system is trained to recognize the speech of a special person; in this case we've a longer training with different speakers that pronounce words in slightly different way. In the case of speaker recognition, we may lie on a fixed text or not.
Regarding **Signature** if we aim to recognize a static signature, we use computer vision techniques so geometrical characteristics of the signature; however if we want to recognize the dynamics of a signature there are for example special pen devices or touch pads that are able to compute the altitude of the pen, the azimuth and also measure the pressure. Each of these elements makes up a time series because while we write, we move the hand and in consequence the pen; so each such elements can be captured in a sequence of timestamps and make up a time series.
Signature is a special case of handwriting. So we can either analyze the aesthetic sign but also the dynamics using a number of time series that are produced by parameters like the altitude of the pen over the plain.
Regarding **Fingerprints**, we've three levels of features. We've a number of pre-processing steps because in principal we've an image but this image is not produced by a vision sensor but produced other kind of sensors; what we've to do is extract the ridges because what we obtained at the beginning is a gray level image. Starting from the input image we can estimate the orientation of the rides and locate the fingerprint; using the orientation field and the region of interest, we can extract the ridges. Given this we can process fingerprints at different levels; we've a global level in which we can notice in the fingerprints whorls, arcs or other traits. Then we've a second local level based on minutiae: minutiae are relevant points in the fingerprints.
Iris has a huge randotypic component so the texture of the iris is mainly randotypic and this is similarly to fingerprints. We've 10 different fingerprints (one for every finger). Even for the iris: the left one is different from the right one.
Another important element to taking into account when talks about iris is that they are usually processed, captured using near to infrared wavelength, because this allows to captured texture of the iris also for very dark eyes.

Since the iris is a circular element is very common to unwrap it in a kind of polar coordinates (it's not exactly a polar mapping because actually this unwrapping takes into account the center of the eye that is conventional identifies as the center of the pupil and then there are some corrections that are applied taking into account the fact that the pupil is not exactly concentric with the respect to the iris and may be we have a kind of effect off axis when the person is looking to a point that is not exactly on the same axis of the point of view of the camera).
The **Retina Scanning** is extremely precise. The blood vessels we've randotypic elements: the pattern of blood vessels, wherever you capture them, are extremely distinguish. So the retina scanning maps the capillary vessels on the eye-ground.
The most huge amount of techniques is devoted to **Face Recognition**. We've a lot of techniques that are image based (that process the image either 2D or 3D like the so called Morphable Model) or we have feature based techniques (in general when we take geometric measures of face's elements and their relationships).

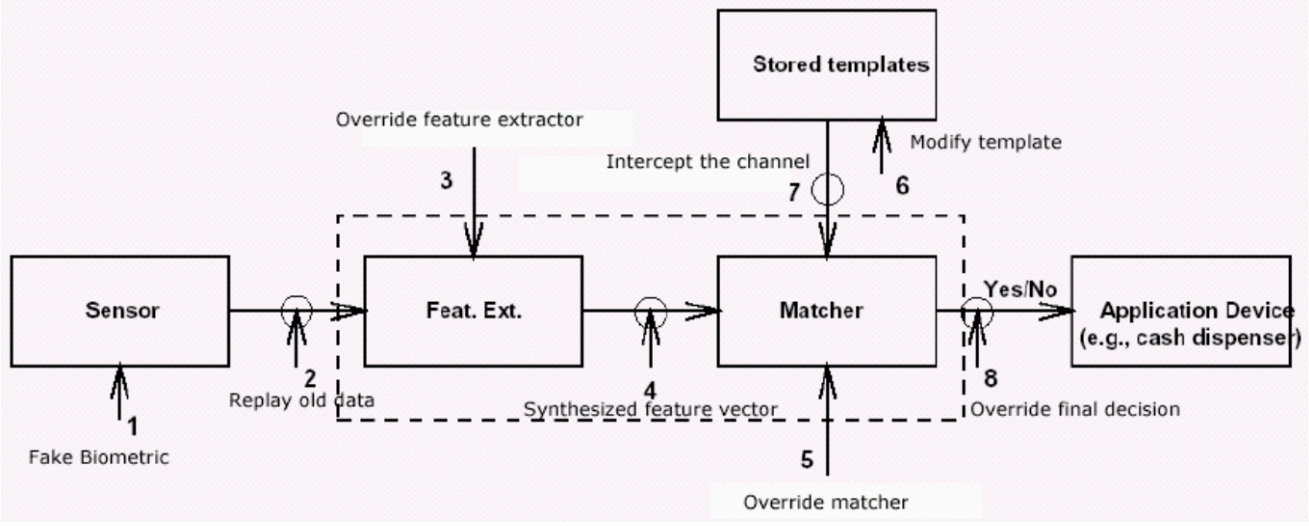
THE USE OF BIOMETRICS TRAITS

Biometric traits are a “natural” authentication methodology. Which are the **benefits** using biometric traits? We don't need to take anything with us, just ourself. We don't need to remember passwords, we don't have to take our credit card with us. So Biometrics traits cannot be lost, lent, stolen or forgotten. The user must only appear in person.
However a photo of our face can be used instead; this is much easier to steal. So spoofing has became a very hot research topic.
Which are the **drawbacks**? We cannot expect in any few controlled situations a 100% accuracy. Some users cannot be recognized by some technologies (e.g. heavy workers show damaged fingerprints). Some traits may change over time (e.g. face).
If a trait is “copied”, the user cannot change it, as it happens for usernames or passwords (plastic surgery). Biometric devices may be unreliable under some circumstances.

PERFORMANCE

We start from the assumption that "all that glitters is not gold": in the sense that, as we've already underlined, biometrics or biometric recognition is not perfect. So there is an amount of flexibility that must be allowed to recognize a person in different settings and in different conditions; but the same flexibility causes possible errors.
Which are the main sources of errors? In order to reliably evaluate a biometric system, we must first of all gain a reliable dataset for testing that somehow resembles as much as possible the real situation where the system will be in operation.
If we consider for example face recognition, we've the so called **intra-class variations**. We've variations within the same class (that is within the set of images relevant to the same subject). Those images never exactly equal to each other.
Intra-class variations include the pose and the expression of the face.
The perfect image must be frontal, with homogeneous illumination, neutral expression. This kind of image is the easiest to recognize, in fact this is usually used (captured) in the case of the controlled conditions.
In the case in which the person wears glasses, the change of pose not only changes some geometrical relationships, but also the effect of glasses changes; this because the reflex and the geometry of the glasses changes with the point of view.
Considering that the expression recognition is one of the hardest task because the expression must be exaggerated in order to be recognize, it has anyway an influence in

the recognition of face. The change of the expression causes the change of the geometrical relationships of the elements of the face.
If the person changes the kind of glasses, the geometry relationships change and there can be different shadows on the face. If the person takes of the glasses, we have another changes. These are all elements that can affect the recognition.
3D models can be less affected by number of condition: for example pose doesn't affect 3D model because we can rotate it and get a different point of view of the same thing. However 3D models are affected by expression as well because even in 3D the expressions can change the geometry of some face elements.
However we got the opposite problem with face recognition: not only huge intra-class variations but also possibly very **small inter-class variations**. They are small variations between different subjects. This is much more frequents with relatives (especially with twins or father and son). In general relatives may cause small problems of inter-class variations: two brothers with a similar age; father and son after a certain age (because of course if the father is sixty and the son is twenty, it's not possible to confuse them). But if they reach an age in which theirs somatic elements become very similar, in particular conditions (for example: when they have similar expression; with a particular illumination), it's difficult for the system distinguish them.
Other problems are given by **noisy and/or distorted acquisitions**. The quality of a sample is extremely important. We can exploit some tricks in order to normalize the illumination. When there isn't an even distribution of illumination in an image, then we can try to normalize, so shadows have much less effect. For example we can also have a poor quality fingerprints (especially with heavy worker because they have damaged fingerprints). We can obtain a bad quality fingerprints also because of dry skin.
Another problem is the **non universality**.
For example, with the fingerprints there is a statistic in which 4% of population presents poor quality fingerprints. This means that this part of population cannot be recognize. Even is only a 4%, according to the extend of the use of application, this can be a problem.



We can also be subjected by **possible attacks (spoofing) in different moments**. Recalling the architecture of a biometric system, we've the sensor where we can put a fake biometric; then the sample is possibly transmitted to the feature extractor module through a kind of communication line (this is the case in which the sensor is not directly connected to the computer); it's possible for an attacker sent again a loaded sample, so a sample that has been already sent in the past and that has been successfully recognized. This can affect the reliability of the recognition.

Then there is a matcher; again, whenever there is a communication line it's possible to inject either a fake sample or a fake vector that is already extracted; it's possible to override the matcher, so that even though the matching result is poor, we can force a higher similarity value for example, in oder to fool the system.
It's possible to attack the archive of stored templates, in order to injecting impostor templates in the archive of the enrolled users for examples.
All these sorts of errors can produce a different effect depending of the kind of application that we use.

POSSIBLE ERRORS: VERIFICATION

Let's start from **Verification**. Verification entails a claim of a identity by the incoming user and the subject is accepted if the similarity (or score) achieved from matching with the gallery template(s) corresponding to the claimed identity is greater than or equal to the acceptance threshold (or, if the distance with such gallery template(s) is less than or equal to the acceptance threshold). Otherwise it is rejected.
What is the main element affecting the outcome of the system, given a certain classification and a certain recognition procedure? What is the main affecting element that must be setup after an accurate testing process? The **threshold**, because it's something that it must be investigated.
According to the measures carried out by the system, we can decide if the person is who claims to be. Whatever the type of criterion, either the similarity or the distance, it's always possible to identify **4 cases**:

- The claimed identity is true and the subject is accepted (**Genuine Acceptance**, GA, also indicated as **Genuine Match**, GM). Positive outcome, because we've actually correctly recognized the person.
- The claimed identity is true but the subject is rejected (**False Rejection**, FR, also indicated as **False Non Match**, FNM, or **type I error**). In this case we've a genuine claim, so the person is who claims to be, but the system doesn't reach a sufficient score, so that either the similarity is to low or the distance is to high with the respect to the claimed identity.
- An impostor subject is rejected (**Genuine Reject**, GR, also indicated as **Genuine Non Match**, GNM). This is another good outcome. The person is claiming a different identity from the true one and the system correctly provides a comparison value that make us reject the claim.
- An impostor subject is accepted (**False Acceptance**, FA, also indicated as **False Match**, FM, or **type II error**). Due to the excessive flexibility that can lead to accepting the person that is not who claims to be.

What is more critical between FA and FR? We mainly focus on False Acceptance when we have security related application; while the False Rejection may only be disturbing and little bit frustrating for the user, the False Acceptance may be dangerous in many situation because, for example, it may allow the access to a terrorist or a criminal.
If we raise alarms for a false rejection, we can create panic, so also FR is important.
In general, FA is the most critical situation but the criticality depends on the kind of applications.
FR creates usability problems for the system because if the rate of false rejections is to high, we may have an unusable system because nobody is allowed to enter.
So we've to reach a compromise between usability and security; it's not possible to decrease both kind of errors or to optimize their behavior at the same time.
When we want to have a kind of error measure that doesn't depend on the size of the probe set, instead of considering absolute values we go to consider rates.

Rates means that we normalize the number of error with the respect to the correct population.
The rate takes into account the absolute number of errors over the number of people in correct population; that is, for the false reject, the genuine people that is correctly claimed their identity, while, for the false accept, is the impostor population.
False Acceptance Rate (FAR) is defined as the percentage of identification instances in which false acceptance occurs. However we've to taking into account the how many imposters submitted their probes to the system; not how many probes we've processed by the system.
This can be expressed as a probability. For example, if the FAR is 0.1 percent, it means that on the average, one out of every 1000 impostors attempting to breach the system will be successful. Stated another way, it means that the probability of an unauthorized person being identified as an authorized person is 0.1 percent.
False Rejection Rate (FRR) is defined as the percentage of identification instances in which false rejection occurs.
This can be expressed as a probability. For example, if the FRR is 0.05 percent, it means that on the average, one out of every 2000 authorized persons attempting to access the system will not be recognized by that system.
False Match and False Non Match count the actual number of errors by the classifier; so they are results by the classifier that doesn't reach a sufficient score to be accepted or go beyond the score being imposters.
On the other hand, False Acceptance Rate and False Match Rate are different: False Acceptance Rate in a real contest or even in a experimental contest where we don't only have the comparison but also a kind of pre-processing of the samples, we also have the kind of error that is **failed to enroll or failed to acquire**. This means that if we have a sample and we're not able to extract a good template, we reject the sample.
So False Acceptance also takes into to account failed to enroll or failed to acquire.
When we want to compare systems, we can somehow look at the curves produced by the False Acceptance Rate and the False Rejection Rate when we move the threshold.
Comparing pairs of curves could be difficult to get a final decision; so we usually rely on single value able to express the overall accuracy of the system.
How to compare system? The most common measures to compare Verification Systems are: plot the curve produced by False Acceptance Rate with the curve produced by the False Rejection Rate; an important operating point is called **Equal Error Rate** (ERR, it is given by that threshold where the FAR is more or less equal to the FRR); while the ERR is a single point, the **Receiving Operating Characteristic Curve (ROC)** plots a kind of general behavior of the system (it has FAR on the x axis and 1-FRR on the y axis; higher is the curve towards the left upper half of the quadrant of the coordinates, better is the system).
It could be difficult to compare systems according to curves, so we have a single value that is valuable in order to compare the systems and it is the area under the ROC curve. We compare two systems just according to the shape of that ROC curve: higher the convexity toward the left upper corner, the higher will be the area of under the curve and better will be the system.
All such measures depend on the threshold. That's why we speak about curves: if we move the threshold, we will get different error rates and they represent points in the Cartesian plan.
When we evaluate a system, we have a **ground truth**; all the data (samples) used for the evaluation experiments is labeled with the correct identity (ground truth).
In general the evaluation is carried out offline; that means that it's not carried out in operational contest, but using a suitable dataset with a reliable ground truth. That is each

template is labeled with the real identity so that in an offline session we can take a group of probes and see what happens during testing the operations.
Higher is the number of the probes that we taking into account, the more generalizable will be the result. Larger is the ground truth, the better will be the evaluation of the system and also a possible training activity.
All the data (samples) used for the evaluation experiments is labeled with the correct identity (ground truth) but this is not true during real world operations; for example, if there is an imposter that presents himself with the so called "**zero effort attack**" (i do nothing to be recognized as another person) there is nothing telling the system whether the claim is true or not so that if we have a false accept, that person will enter.
Generalizability is important: we try to use the larger possible dataset as ground truth, that is somehow similar in conditions with the real samples that we'll have to process during the real operations.
In general we assume to have a **ground truth function id(template)** that, given a template as parameter, returns its true identity, for instance:
- $id(p_j)$ is the true identity associated with the j-th probe
- $id(g_x)$ is the true identity associated with template labeled as x in the gallery
- i is the identity claimed by a probe p_j

We can do this because during testing we always know which is the true identity. So this is strictly limited to testing.
We have a function called **topMatch(pj, identity)** that returns the best match between p_j and the (possibly more than one) templates associated to the claimed identity in the gallery.
So 'identity' in topMatch is the claimed identity; pj is the submitted probe; topMatch is the best match.
If we have a single template per identity in the gallery, topMatch just will be the distance between pj and such template.
The function $\mathbf{s(t_1, t_2)}$ returns the similarity between template t_1 and template t_2.
Is strictly required to have no overlap among the sets that we use during the training or testing.
On the other hand, if we have more templates for the same identity, we choose the best template (if we use the distance, we choose the template with the lower distance) and we associate this measure to topMatch and then we compare that with the threshold.
This is not the only possible strategy, because in general when we have more templates we choose the one with the lower distance, but we can choose also to return an average.
Return an average however will somehow decrease the advantage to have more templates in the gallery.
We have the **Gallery**, that is the **set of the enrolled templates** (the people that are allowed to the system).
We can have an enrolled person that declares false identity. Why could this happens? If there are people with administrator rights (people whose granted to access to restricted areas) and so they can do tasks that are not allowed to normal users, even an user that is enrolled as normal user could try to access to the system as administrator. In verification, an impostor is not necessary a person that is not enrolled in the gallery; but just a person that claims a false identity.
In general we divide our dataset into probe and gallery. **What happens to the probe set?**
The probe set can contains a **subset $\mathbf{P_G}$** , which is the set of probes belonging to subjects in the gallery (genuine claims can only come from here), and a **subset $\mathbf{P_N}$**, which is the set of probes belonging to subjects not in the gallery (but in the dataset, so id function works).

The sets of subjects in the gallery/not in the gallery is decided during experiment set up but all samples/templates are labeled in any case.
Also the persons in the gallery could claim false identities.
FRR is always computed for people in P_G; a FR can only happen for people that are in the gallery and that are not recognized.
So we have the number of false rejects divided by the cardinality of the set of people that we must correctly consider in order to compute this rate.
The number of false rejects is given by the cardinality of the set of all those probes p_j such that the similarity between the template x in the gallery and the probe j is below or equal the threshold and the identity of the user of the template x in the gallery is exactly the same identity of the user of p_j.

$$FRR(t) = \frac{|\{p_j : s_{xj} \leq t,\ id(g_x) = id(p_j)\}|}{|\{p_j : id(g_x) = id(p_j)\}|}$$

When the user submitting the probe sample (template) p_j declares the true identity, then such identity will be the same returned by the ground truth function for p_j that is $id(p_j)$.
The s_{xj} that we find in the formula, usually correspond to the topMatch between the claimed identity and templates belonging to that identity.
g_x is the template achieving the topMatch between the probe pj and the templates that corresponding to its identity. We can say that they're corresponding to its identity because we assume that the claim is genuine.
s_{xj} is the similarity between g_x and p_j and in this case we can measure it for each probe belonging to the set P_G.
We use $id(p_j)$ instead of a generic i to underline that the claim is genuine.
What about the FAR? We have the cardinality of the set of probes p_j such that the similarity between the probe and the claimed identity is higher than the threshold but the claimed identity is different from the true one.

$$FAR(t) = \frac{|\{p_j : s_{xj} \geq t \ \wedge \ id(g_x) \neq id(p_j)\}|}{|\{p_j : id(g_x) \neq id(p_j)\}|}$$

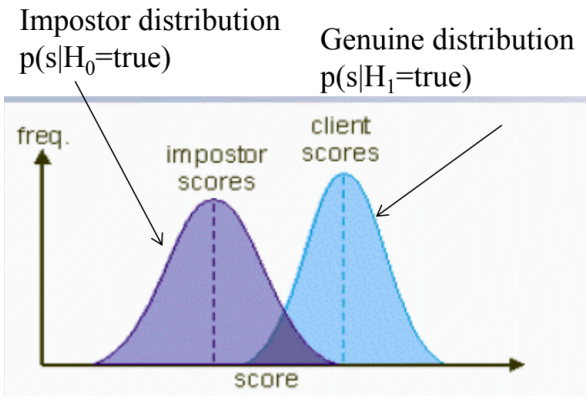
s_{xj} in this case is simply a similarity measure; we can substitute distance d to s, just to underline that we're using the distance. In these formulas we're using the similarity.
FRR is computed against the number of genuine claims, while FAR is computed against the set of false claims.
When the user submitting the probe sample (template) p_j declares a false identity, then such identity will be a generic identity i which is different from that returned by the ground truth function for p_j that is $id(p_j)$.
We have **two scenarios** in this case: in one scenario we can have p_j belonging to all those subjects that are not enrolled; in the other scenario we can have p_j belonging P_G union P_N.
It can be considered a "for each" only assuming that all genuine also claim a wrong identity. We take p_j from P_G union P_N just in order to take into account that also a genuine user can declare a wrong identity.
From the point of view of the computation, the fact that the identity is either enrolled or not is, is not relevant for the computation; what is relevant is the genuinely of the claim.
Of course a non enrolled person can only claim a false identity; the difference is given by genuinely. In a simply scenario, enrolled people can only claim genuine identity; in a more

complex scenario, we may also consider that an enrolled person can also declare a false identity.
The two scenarios differ only for the fact that the impostor can either ALSO belong to the gallery (a registered subject, whose probe template is therefore in P_G, but declaring another identity) or not (a subject who is not even registered, whose probe template is in P_N).
The difference is not important for the computation, since in both cases we have a false claim.
What is really relevant to compute the two rates is the reference set: in one case we have the number of genuine claims; in the other case we have the number of false claims.
False Rejection Rate is complementary to Genuine Acceptance; so FRR + GAR sum up to 1, because they are exactly complementary. The same happens for False Acceptance Rate and Genuine Rejection Rate, because both rates are measured against then number of false claims.
We can somehow put in relation **two different hypothesis**: in $\mathbf{H_0}$ the person is a different person of the claim; in $\mathbf{H_1}$ the person is the same person as the claimed identity. The **possible decisions** are: $\mathbf{D_0}$ declares that they are different persons; $\mathbf{D_1}$ declares that they are the same person.
These decisions are correct only when we take into account the hypothesis.
FAR is the probability of having a D_1 decision in the presence of H_0 true hypothesis. It means that it's true that we have a different person incoming and then the conditioned probability of providing an accept decision when we have a different person incoming is FAR from a probabilistic point of view.

$$FAR = p(D_1|H_0=true)$$

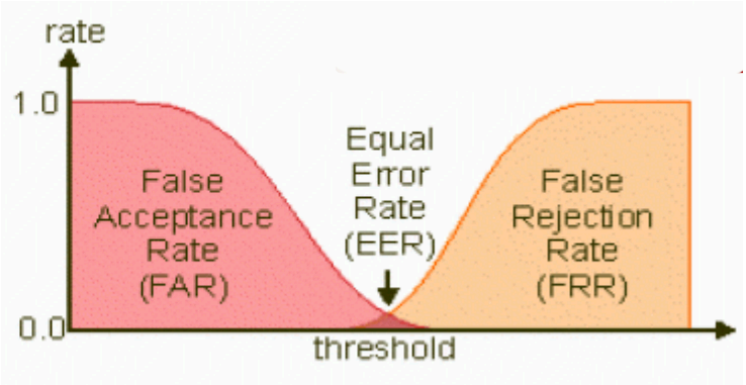
$$FRR = p(D_0|H_1=true)$$

The FRR is the probability of providing the response that declares the person are different, given the H_1 hypothesis instead. That is: we have the same person as the claimed identity, but the system provides a different person response.

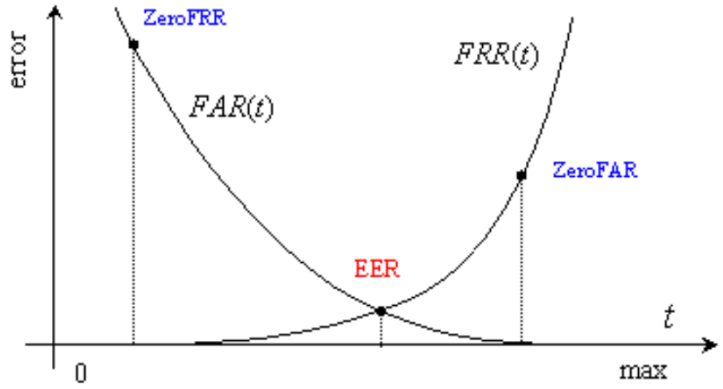


In this plot we are depicting the **Impostor Distribution**: probability of s, where s is a similarity value, given that the hypothesis H0 is true.
We can assume that the distribution of the probability is a normal distribution, so that most similarity scores achieved by impostors will concentrate around a certain average value and then, according to the kind of similarity that we use, we may have a larger gaussian curve or a smaller one.
On the other hand we've the **Client Scores**, since we are taking into account similarities; such similarity scores will move in the upper part of the x axis, so that in general we can expect to higher Client Scores with the respect to the impostor ones.
A score is said **genuine** (authentic) if it results from matching two samples of the biometric trait of a same enrolled individual; it is said **impostor** if it results from matching the sample of a non-enrolled individual.
Where does the problem rise? The problem rise in the intersection of the two diagram distributions, in the sense that there is a part of Impostor Scores that can fall in the same region of the Client Scores. If the person is an impostor and we obtain a score that is in the intersection, according to the threshold that we've setup, we could have a false

accept. In the same way, if we've a genuine user that achieved a similarity score that is towards the Impostor Scores distribution, again, according to the threshold, we may have a false rejection.
If we put the threshold where the impostor distribution ends, we can say that we only accept similarity higher than that point. We've to look how many genuine users remain excluded from the system. So it's not possible to have this kind of decision, unless we're looking for a very high security level.



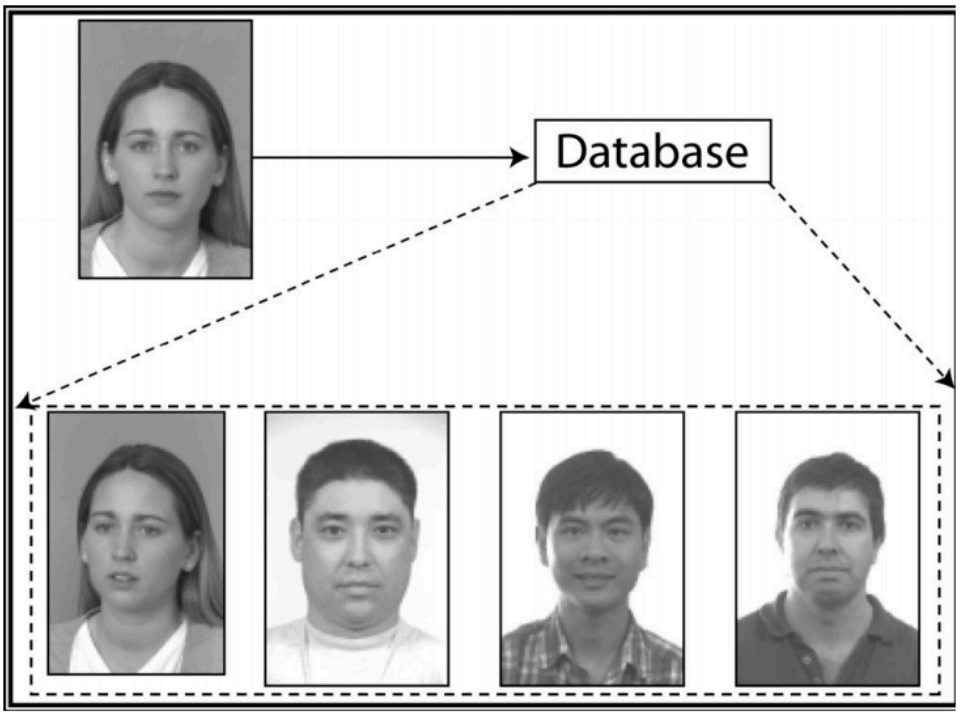
In this diagram, we've the False Acceptance Rate curve and the False Rejection Rate curve.
FAR and FRR have opposite trends with the respect to the threshold. If we increase the threshold, we decrease the FAR.
We can interchange the role of similarity and distance and choose the most suitable measure that we want to exploit.
As we increase the requested similarity, we will decrease the FAR. On the other hand, when we further increase the similarity requested between the probe and the gallery templates, we increase the possibility to issue FRR. If we consider distances instead of similarities, in the previous diagram we have to invert the two distributions (Impostor Scores and Client Scores), because a lower distances will be correspond to Client Scores. Even the two curves relative to FAR and FRR will have an opposite meaning because once we increase the allowed distance, we will decrease the FRR but increase the FAR.
The point provided by that threshold where FAR and FRR achieved a similar value is called **Equal Error Rate** (**EER**). The EER is not a threshold but it's the value that we achieved setting up a similarity threshold where the FAR is equal to the FRR.



Together with the ERR, we can consider other kinds of parameters that can be used when we have specific requirements from the system. We can use for example the so called **ZeroFAR Point**, that is better defined as **Zero False Match Rate**. It's the value of FRR when we reach a 0 FAR.
The same happens on the other hand: we have **ZeroFRR** (**Zero False Non Match Rate**) that is the FAR when the FRR is equal to 0.
We could never reach the condition where we have a FAR exactly equal to 0 ora a FRR exactly equal to 0. So these are somehow some conceptual points that can be used.

Acceptance threshold is crucial and depends from the application needs.
The Acceptance threshold cannot easily changed when the system is in operation, so it's one of the elements that has to be accurately decided during the evaluation.
Considering distances, in the last diagram we have two curves with opposite trend: below the distance threshold we've the Biometric Feature Accepted while if we go above the threshold we've the Biometric Feature Rejected. So when the biometric feature is accepted we can have a False Acceptance, otherwise we've the False Rejections.
The most curve used is the **Receiver Operating Characteristic** (**ROC**) curve.
ROC depicts the probability of Genuine Accept Rate (GAR) of the system, expressed as 1-FRR, against False Accept Rate (FAR) variation. We've a range from 0 to 1; on the x axis we've the FAR and we look what happens when we've a certain False Acceptance Rate to the Genuine Acceptance.
If we reason about the meaning of this curve, we may understand that the higher the corner of the curve towards the upper left corner of the plane, the better is the curve.
Another kind of curve that is used is the **Detection Error TradeOff** (**DET**). The DET curve depicts the probability of False Reject (FRR) of the system, against False Accept Rate (FAR) variation. It is plotted in logarithmic form.
In the DET curve we compare the two kinds of error directly. The meaning of the corner of the curve in this case changes: lower the curve is, better is the curve.
Now we're looking for the **possible errors when we have Identification open set**.
Identification open set is also called sometime "watchlist", despite they are not the same. Watchlist entails have either a blacklist or a whitelist; in general Identification open set gives no weight very to the fact that the person belongs or not to the list, but it's simply an enrolled person.
In the open set identification task the biometric system determines if the individual's biometric signature matches a biometric signature of someone in the gallery.
In the open set identification, the biometric system determines first whether the individual is enrolled in the gallery. So first of all the first aim is to see if this person is in the gallery. After that we want also to get the identity of that person. The difference with the verification is that the individual doesn't make any identity claim.
In this case we can have **more possible errors situations**, because they depending on the matcher and on the recognition threshold (score/similarity/distance).
What are there more possible error situations? In this case the threshold mainly determines whether the incoming probe is recognized as a probe belonging to the gallery. So in this case, since we've not a claimed identity but we've a matching with an entire gallery. The threshold is just a way to detect the presence of the person in the gallery.
If we've a similarity threshold, if the similarity doesn't reach the threshold, this means that this person is not similarly enough to any template in the gallery. So this means that this person doesn't belong to the enrolled subjects.
So the role of the threshold is to decide whether this person can be considered as an enrolled person or not, only if at least one comparison matching value is above the similarity threshold.
Since we've to have carrying out 1 to n comparisons, so we're compering the probe with all the templates of all identities in the gallery, whatever is the value that will return, we've a list of such values that is ordered according to the criteria that we've adopted. In the sense that if we've for example similarities, we've a list of values in decreasing order.
So we have this list of values and the threshold; the list of similarity values is not affected by the threshold (the similarity is the same whatever is the threshold; the similarity can only change when we change the matcher).
However as we move the threshold along such list we may find different situations. For example, if not even the better value that is the first one is over the requested similarity,

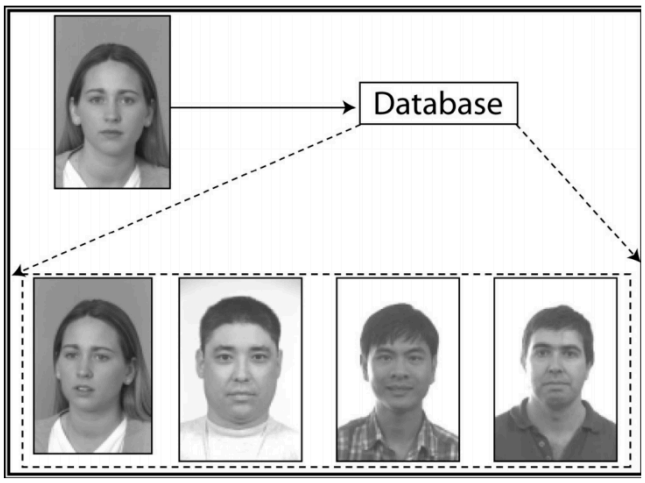
this is the case when the threshold suggest that this person is not an enrolled person (because not even the best similarity value is equal or above the threshold).
On the other hand, we may have a variable number of values above the threshold.
What happens when we don't have values above the threshold? We may have genuine rejects (because the person may not really be in the gallery; in this case we cannot speak about real impostors, because the person doesn't claim anything) but if the probe belongs to a person in the gallery and non templates in the gallery achieved a sufficient similarity, we have false rejection.
If we use distances instead of similarities, everything must be inverted.
What happens when we have similarities above the threshold? This means that we have detected that the probe is in the gallery.
Considering that we may have more than one value above the threshold, we consider the first one because in general even a identification system just returns a single identity.
If the first one corresponds to the true identity of the probe, we've a correct identification (correct detection end identification). We've two optimal results: we successfully detected that the person was in the gallery and we also correctly identified it.
However, it may also happen that the first identity in the list is not the correct identity. Even though the person is in the gallery (so we've a correct detection), the first returned value doesn't correspond to the correct identity (this correspond to a false rejection from the point of view of the correct identity). So in identification open set, we've **False Rejection** in two cases: either when there is not detection at all (the similarities are below the threshold) but also when the first returned identity meeting the threshold it is not the correct one.
Finally we've a **False Acceptance** when whatever is the first identity in the list, the probe of a person who is not in the gallery achieved a similarity value above the threshold. In this case, we don't care about the list, because the person is not in the gallery, so there is not correct identity.



Scores = 0.9 0.86 0.6 0.4

Threshold = 0.85

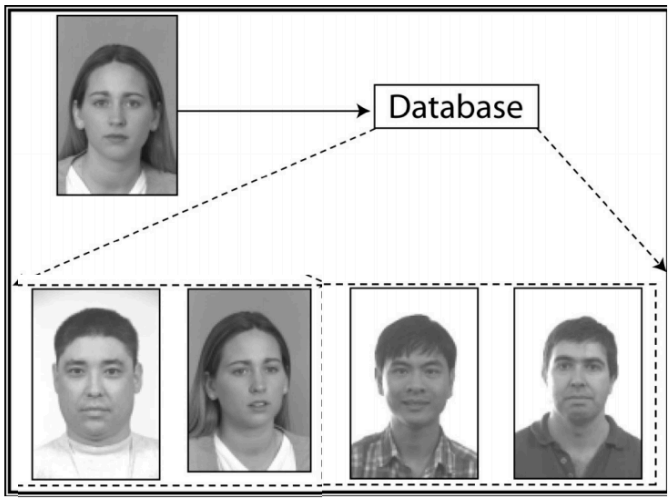
In this case only two individuals are above the threshold and we have a **correct detect** or, the so called, **genuine alarm**. We correctly detect that the probe belongs to an individual in the gallery. Moreover, in addition to this, the first individual is the right one, so we also have a **correct identification**.
This outcome is defined as **“correct detect and identify”**.



Scores = 0.9 0.86 0.6 0.4

Threshold = 0.95

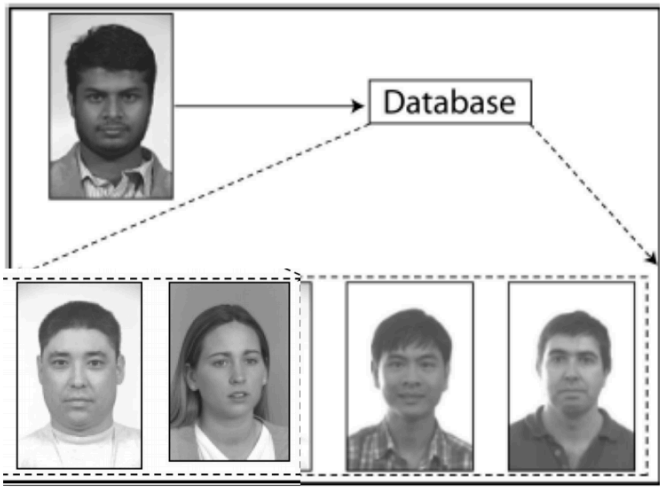
In this other example, no individual is above the threshold, but from our ground truth (considering that during the evaluation we always know which is the correct value) we know that the person is in the database (**no correct detect**). In this case, we don't care the order of the list returned by the system, because the only relevant event is that all the similarities are below the threshold (**no correct identification**).
Se we have a **"no correct detect and identify”**.
In the following example, we’ve two individuals above the threshold, so we have a correct detect alarm, but the first individual is not the right one, so we have again **"no correct detect and identify”**.



Scores = 0.86 0.8 0.6 0.4

Threshold = 0.75

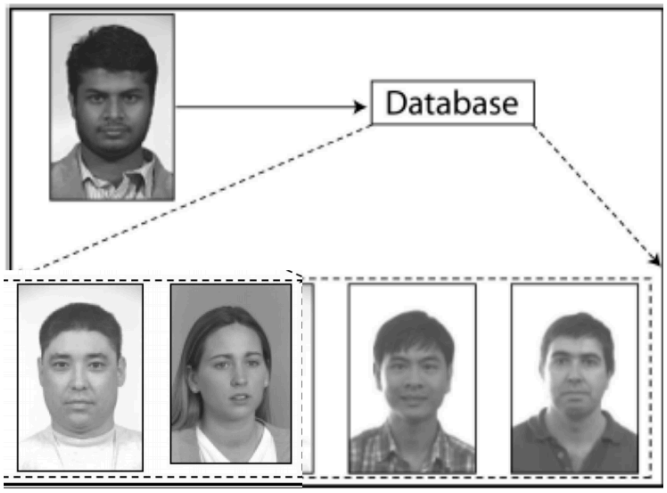
In particular, in this case, what is missing is the correct identification.
If we run many trials with probes belonging to the subjects in the database (set P_G), we will know how often the system will return a correct result.
A correct result occurs when the individual in the probe image is also in the database AND the correct individual has the highest similarity score.
This is called the **correct detect and identify rate**.



Scores = 0.8 0.7 0.6 0.4

Threshold = 0.85

This is an example where the probe doesn't belong to an individual in the database. There is no individual above the threshold (no detection), so we don't care about looking at the top individual (no identification). So we have a **genuine reject** (correct result).



Scores = 0.8 0.7 0.6 0.4

Threshold = 0.75

With a lower threshold, we've one individual above the threshold (detection). Having only one individual above the similarity threshold, we don't care about the remaining order; the important thing is that we've a **False Acceptance** because this person, whatever is the returned identity, was not in the gallery. We have a **False Alarm**.
If we run many trials with probes belonging to subjects not in the database (set P_N), we will know how often the system will return an incorrect alarm.
This is called **false alarm rate**.

When we have a correct detection, we can also consider the position in the list where the first template for the correct identity is returned (because the correct identity may not be in the first position). So we can consider its **rank**, that is the position in the list of the template belonging to the correct identity.
Detection and Identification Rate (DIR) at rank k, measure the the probability of correct identification at rank k (the correct subject is returned at position k). Given that the first k positions are all above the threshold. Otherwise we have False Rejections.
The rate between the number of individuals correctly recognized at rank k and the number of probes belonging to individuals in P_G can be defined in this way:

$$DIR(t,\ k)\ =\ \frac{|\{p_j :\ rango(p_j) \leq k,\ s_{ij} \geq t,\ id(g_i) = id(p_j)\}|}{|P_G|}\ \forall p_j \in P_G$$

On the dominator we have cardinality of the set P_G, so the set of probes belonging to the gallery. Then we have the number of probes from which the rank is lower than k, the similarity is higher than the threshold and the identity is correct.
The probability of false reject expressed as 1-DIR(at rank 1), so **FRR(t) = 1 - DIR(t, 1)**.
Detection and Identification Rate is more or less equivalent to Genuine Acceptance; False Rejection is the complement of the GAR, so we can compute FRR just doing: 1 - DIR(at rank 1).
We can also define False Acceptance Rate, so in this case we don't care about the position in the list; we only consider that we have a probe not in the gallery for which the highest similarity achieved is above the similarity threshold.
So the **FAR** is the rate between the number of impostor recognized by error and the total number of impostors in P_N.

$$FAR(t)\ =\ \frac{|\{p_j :\ max_i\ s_{ij} \geq t\}|}{|P_N|}\ \forall p_j \in P_N\ \forall g_i \in G$$

Again, we can define the **Equal Error Rate**: is provided by that threshold for which we have the same value for FRR and FAR, so it's the point where the two probability errors are equal.

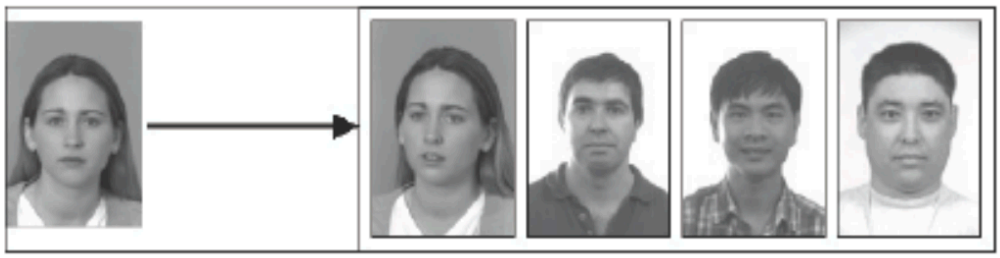
$$EER\ =\ \{x :\ FRR(t) = x\ \wedge\ FAR(t) = x\}$$

As in verification, we would like to be able to set our threshold so that the detect and identify rate is 100%, and the false alarm rate is 0%.
This is not possible for the same reasons : FAR and FRR both depend from the score/ similarity/distance threshold (they are connected) yet in opposite directions: if we raise the threshold, the detect and identify rate decreases, but our false alarm rate also decreases; if we lower the threshold, the detect and identify rate increases, but our false alarm rate also increases.
We can plot detect and identify rates and their associated false alarm rates.
We can call this plot Open-set (Watchlist) Receiver Operating Characteristic, or Open-set (Watchlist) ROC.

Selection of a watchlist threshold will depend on the kind of application.
In practice, we can identify five operational areas:

1. **Applications requiring extremely low false alarm.** When any alarm requires immediate action, this could lead to public disturbance and confusion. Moreover, an alarm and subsequent action may make evident that surveillance is being performed and how, and may minimize the possibility of catching a future suspect.
2. **Applications requiring extremely high probability of detect and identify.** The main concern is detecting someone on the watchlist; false alarms are a secondary concern and will be dealt with according to predefined procedures.
3. **Applications requiring low false alarm and detect/identify.** The main concern is lower false alarms and it is acceptable to deal with low detect/identify.
4. **Applications requiring high false alarm and detect/identify.** The main concern is higher detect/identify performance and it is acceptable deal with a high false alarm rate as well.
5. **Applications requiring no threshold.** The user wants all results with corresponding confidence measures for investigation.

Closed set identification is a special case of the open set identification where we assume that each and any probe belongs to an enrolled subject (so this is not realistic).
In the identification closed set **we don't have threshold at all**; so the only possible question that we ask to such system is: **who is the person submitted the probe?**
The only kind of error that we can make is to return the wrong identity, because that person will be surely in the list.
So there is just a kind of False Rejection (when we return the right identity in a position that is not the first one) and there is no False Acceptance.
In practice, there are very few applications that operate under the closed set identification task.
Identification Closed Set is not a really realistic setting but is useful in order to evaluate the performance of the system, just because the evaluation of Identification Open Set is more difficult to compute.

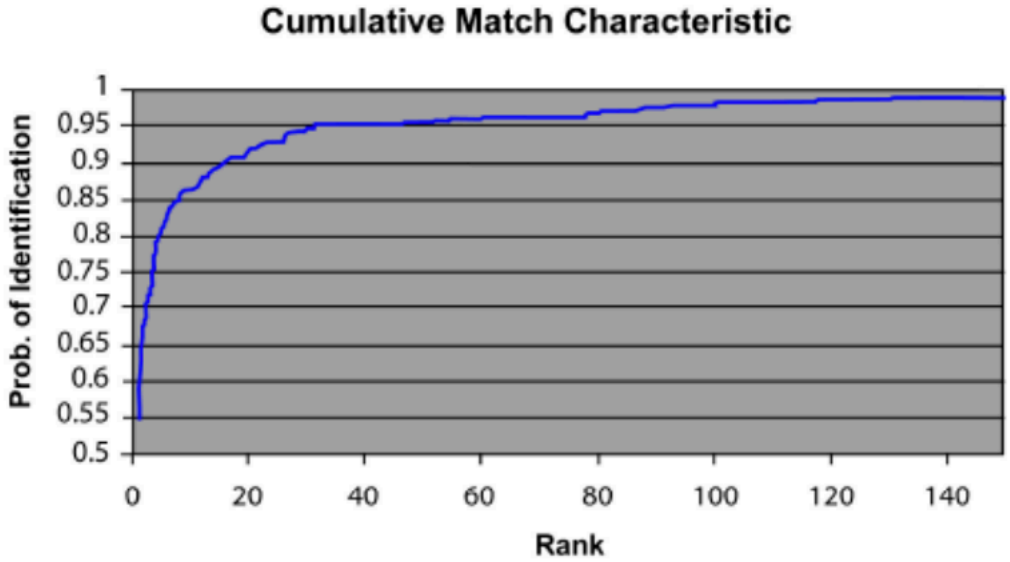


Scores= 0.9 0.86 0.6 0.4

In this example, the correct match has the top similarity score.
If we run many trials with different subjects, we will know how often the system will return a correct result with the top match.
This is termed the **probability of identification at rank 1**.

Scores= 0.86 0.8 0.6 0.4

In this example, the correct match has the second highest similarity score.
If we run many trials with different subjects, we will know how often the system will return a correct result with either the top or second similarity score (we do not necessarily care if they are in the top or second, just that they are in one of those positions).
This is termed the **probability of identification at rank 2**.
The probability of correct identification at rank 20 means: what is the probability that the correct match is somewhere in the top 20 similarity scores?
How can we provide the results of evaluation of Identification system in Closed Set?
We create the so called **Cumulative Match Characteristic curve** (**CMC curve**). The probability of having the correct identification within the k position is called **Cumulative Match Score at rank k (CMS)**.
It therefore reports the probability that the correct identity is returned at the first place in the ordered list (CMS at rank 1), or at the first or second place (CMS at rank 2), or in general among the first k places (CMS at rank k). If the number n of ranks in the curve equals the size of the gallery, we will surely have a probability value of 1 at point n.
So the Cumulative Match Characteristic curve plots the values of CMS for all possible ranks. In general, all possible ranks means that we can arrive to compute the probability that correct identity is within the end of the list so we surely reach probability 1. Because since we assume that all people is in the gallery, at least we will have the correct identity as the last one. So CMC curve always reaches the value of 1.



Higher is the area under the curve, the better is the system.
The x axis depends on the dimension of the gallery, so we can't normalize or use it in 0 and 1.
The maximum area is equal to the size of the gallery because it's always the size of the gallery multiply by 1.

In order to have a normalized value, we could divide the actual area by its maximum; so we can have a value between 0 and 1.
What we're most interested in is CMS at rank 1, that means what is the probability to get the correct identity exactly in the first position.
This special values is also known as **Recognition Rate**.
Recognition Rate can achieved a certain value so it could not be sufficient to express the overall trend of the curve that could remain quite low up to the end; so we can use two farther values that are CMS at rank 5 and CMS at rank 10. After 10 we think that the system doesn't behaving so well.

OFFLINE PERFORMANCE EVALUATION

The statistical (offline) analysis is carried out on suitable «static» datasets, collected to evaluate approaches to a problem, and entails a ground truth.
So, what we always taking into account is that when we carry out the statistical analysis of a biometric system we're working offline and with a suitable ground truth. This means that we don't have any time constraints and all samples are labeled (in the sense that we know their real identities). This is not true in the real operations, where it could not be always possible to check the real identity of a probe, that's why it is important to have a reliable offline testing of the system accuracy to evaluate its reliability.
During performance evaluation, each sample presented as a probe may play the role of either a genuine sample or of an impostor one (never in the same moment), according to the gallery setup for the experiment at hand and on the possible identity claim attributed to the probe if in verification mode.
We can divide our dataset in such a way to carry out as many experiments as possible.
In a typical dataset we've a number of templates for each identity: so we have the same person possibly in many different situations and we've a way to assign the same label to all templates that belong to the same person (it's not mandatory to have the same number of templates for each identity).
The first choice is to **divide the dataset between training and testing (TR vs TS)**. In general training is specially needed when we're working with machine learning approach that requires a specific creation of the model; in order to create such model, we've to carry out training.
The training set and the testing set must never be overlap: training samples must be separated by the testing samples.
When we choose the training set, templates of different quality must be included in the training set, because it must include as many as possible different conditions that will be found in the testing set or in the real life.
Generalizability of the training outcomes depends on the choice of the training set. The generalizability of outcomes, that is the possibility to get the same kind of performance even with unknown data, depends on the choice of the training set.
Some datasets suggest this division in order to assure fair comparison of methods. Since train is important for generalizability, if different methods use different training set, they could get different performance just for this reason.
Recognition methods may not require training in the strict sense (machine learning) but this can be used to set up parameters for testing (not the threshold because it is studied in extensive way, but for example the number of points that must match in a fingerprint).
Choice based on both subjects (a subject may not belong to the training set, to better test generalizability) and on samples (no overlap between TR and TS allowed). The choice how to partition the dataset can be either based on subjects (this means that we're not creating a model for each subject; however we may also have some subjects in the

training set, so that we create the appropriate model, and other subjects that not appear in the training set, the so called the never seen subjects, just to see how the system reacts to these subjects) or base in samples (we can have part of templates of subjects in the training set and part of templates for the same subjects in the testing set). What is important is that, from the point of view of samples, no samples has never to appear both in training and testing.
The second choice is to divide the **dataset in probe set and gallery set** (P vs G).
According to the kind of samples that we put in the gallery, we may have very different performance. For example, one choice can be based on choosing the best samples to be in the gallery and more samples in more different conditions as probes. This choice is reasonable, because in general enrolling into the gallery is carried out in more controlled conditions (when we're registering in the system, it takes neutral photos) while in the photos that will are submitted to a video surveillance system, our face will be captured in very different positions.
However there is another possibility: we try to have template samples with very different conditions in the gallery in order to be able to recognize the same person in different conditions (with and without glasses, with different expressions, and so on and so forth).
In this case the choice is always based on samples, in the sense that no overlap must happen between probe and gallery.
The division in gallery and probe is mostly used for testing.
The third choice pertains the division between open set and closed set, so we can either choice the templates in the probe set always comes from subjects in the gallery (this can be used for identification closed set, but also for verification) or we can have the probe set also including subjects that are not in the gallery (so that P is actually the union of P_G and P_N). This is the case when we have identification open set.
This choice doesn't affect verification because in verification we just have a comparison with the claimed identity whatever is the condition of the probe.
In identification closed set this choice is not possible (non sense).
In identification open set this choice may influence the results according to the ratio among the number of subjects in the gallery (e.g., black or white lists may contain few subjects) and the number of total probes (e.g., all the passengers accessing an airport).
This choice is only based on the subjects (either we deiced the probe subject is in the gallery or not).
In principle, especially when training is involved, we should partition the dataset in different ways, repeat the evaluation, and take average performance, in order to avoid the bias due to a specific choice of partition elements.
So in **K Fold cross validation**, the data is divided into k subsets.
Now the holdout method is repeated k times, such that each time, one of the k subsets is used as the test set/ validation set and the other k-1 subsets are put together to form a training set. The error estimation is averaged over all k trials to get total effectiveness of our model. As can be seen, every data point gets to be in a validation set exactly once, and gets to be in a training set k-1 times. This significantly reduces bias as we are using most of the data for fitting, and also significantly reduces variance as most of the data is also being used in validation set. Interchanging the training and test sets also adds to the effectiveness of this method. As a general rule and empirical evidence, K = 5 or 10 is generally preferred, but nothing’s fixed and it can take any value.

ALL AGAINST ALL

One of the possible strategy in order to increase the number of experiments is to apply an **All Against All computation**, where All Against All means that all templates in the probe against all templates in the gallery. In any case we may have different choices: again probe and gallery, but what is change is that for each probe we can simulate more experiments, because in general we assume for each operation in real world, for example in verification application, each probe only claims one identity.
When we want to massively test our system, we can do something better: we may simulate many tests using the same probe so that, in turn, each probe may be assumed to claim a different identity.
In the following we will consider distance measures (e.g., Euclidean) since similarity entails exactly symmetric considerations.
For each probe and gallery pair, it's possible to compute beforehand a **$\text{ALL}_{\text{PROBE}}$-against-$\text{ALL}_{\text{GALLERY}}$ distance matrix**. This is always possible because we've samples and the ground truth (the correct labels) in advance. It's possible feasible to match in advance each probe with each template in the gallery and to store such distances or similarities in a matrix that can be later used to in order to compute performance measures.
The distance matrix can be used for performance evaluation of all kinds of applications (verification, identification closed set, identification open set, considering both single and multiple templates per subject in the gallery).
Each row may correspond more than one recognition operation on an incoming probe.
Each probe/gallery partition produces a different distance matrix, because if we change the templates that we're going to match the distance matrix changes.
To allow a better generalization, the distance matrices used in the following examples will not contain the numeric values but only their ascending order: higher is the value in the distance matrix, higher must be the acceptance threshold in order to support a positive decision.
For the **Verification** only gallery templates belonging to the claimed identity are matched against the probe. It's not important who is in the gallery, but the claimed identity.
Each row is labeled with the ground truth probe identity and, for verification, with the claimed identity.
In genuine matches the probe is associated with the claims of the correct identity; in impostor matches the probe is associated with the claim of a false identity (wether the subject is in the gallery or not).
Performance evaluation can be obtained by separating probe and gallery (different subsets of templates) and also if we want to have a one-shot experiment we can also divide the probe in genuine and impostors (templates that play the role of impostors are associated with the claim of a different identity).

Example:

Identities A, B, C, D, E, F are in the dataset
Identities A, B, C, D are in the gallery with a single instance (one single template in the gallery for each identity; we may also have much more template in the gallery for each identity, but in that case we take the best result for that identity)
Identities E and F play the role of impostors in all cases
d() = function that measures distance form the probe
t = acceptance threshold

We have only the order shown: in this case each row is a single verification operation and, whatever is the real distance measure, the rank is equivalent to a kind of score. In this case we can setup the threshold according to the rank achieved by the score achieved in the comparison. We setup this ranking in the beginning, so we setup score in relation to the relative ordering of similarities. The real distances are substituted with ordinal numbers, just to denote that distance 1 is lower than distance 2, distance 2 is lower than distance 3, and so on and so forth.

	Probes	A1	B1	C1	D1
ID A – Claim A	P1	1	4	2	3
ID D – Claim C	P2	4	1	3	2
ID E – Claim D	P3	4	2	1	3
ID C – Claim C	P4	…	…	…	…
ID F – Claim B	P5	…	…	…	…

The identity A claim A, so this is a genuine claim. For the first probe P1 we observe that we have A1 that achieves the lower distance, B1 that achieves distance 4, C1 that achieves distance 2 and D1 that achieves distance 3.
In this case we know that probe P1 has identity A and has claimed identity A, so it must be a Genuine probe and should be accepted by the system. If we have a distance lower or equal than the threshold, then we will have a genuine acceptance, so we increase the number of GA.
If we have a distance that is higher than the threshold, then we've to increase the number of False Reject.
Let's go to P2: the probe P2 belongs to the identity D, but the identity claim is C. D is in the gallery but claims a different identity (even an enrolled subject can claim a different identity). So In any case P2 is an impostor.
In this case the lower distance is achieved in comparison with the template that belonging to the identity B. According to the threshold, if the distance falls below the threshold, then we will have a False Acceptance; if the distance falls above the threshold, then we will have a Genuine Reject.
Consider now P3: the probe P3 belongs to the identity E, that it's not in the gallery, so also in this case the probe is an impostor. P3 claims the identity C.
So also in this case we can have either a False Acceptance or a Genuine Rejection according to the threshold.

Example:

Identities A, B, C, D, E, F are in the dataset

Identities A, B, C, D are in the gallery with multiple instances (for each subjects we have more templates in the gallery and we apply the strategy of choosing the best result; that is we use the lower distance between the probe and the claimed identity in order to determine the response)

Identities E and F play the role of impostors in all cases

d() = function that measures distance form the probe

t = acceptance threshold

Again, each row is a single verification operation. In this example we have two values for distances because we have two template in the gallery for each identity.

	Probes	A1	A2	B1	B2	C1	C2	D1	D2
ID A – Claim A	P1	2	1	8	4	3	7	5	6
ID D – Claim C	P2	7	6	2	1	5	8	3	4
ID E – Claim D	P3	7	5	2	8	6	1	3	4
ID C – Claim C	P4	…	…	…	…	…	…	…	…
ID F – Claim B	P5	…	…	…	…	…	…	…	…

Let's start from P1. P1 is a genuine claim. We take the best value from the comparison: in this case the distance value between the probe P1 and A1 is 2, while the distance value between P1 and A2 is 1. So we use the value coming from A2, because it's the best match for claim A.

If the distance d(A2) is lower than the threshold, we've a Genuine Acceptance (we can do that thanks to the ground truth); otherwise we've a False Rejection.

P2 and P3 are both impostors claims. In the case of P2, D belongs to the gallery but claims a different identity, while in the case of P3, E is not a subject in the gallery.

Regarding P2 the claim is the identity C. The distances with C are 5 and 8; the best match for this claim is C1, so we choose it. If the distance d(C1) is lower than the threshold, we will have a False Acceptance; otherwise we will have a Genuine Rejection.

Instead as regards P3, the identity claimed is identity D. The best match in this case is D1 with the distance value equal to 3. If this value is lower than the threshold, we will have a False Acceptance; otherwise we will have a Genuine Rejection.

Having more samples per subjects decreases FRR (because we may have more possibilities to recognize the genuine identity), **but may also increase FAR** (more possibilities for an impostor to look similar to a genuine).
This approach requires to carry out a sufficient number of evaluations in order to compute a reliable average result:

- each time probe set and gallery sets are chosen in a different way
- each time there is a possibly different distribution of genuine probes (the claimed identity is the true one) and impostor probes (the claimed identity of the probe is not the true one, whether the probe belongs to a gallery subject or not), i.e., how many genuine and how many impostors are considered

The choice of genuine and impostors can influence the outcome.

In **Identification Open Set** we may use the same matrixes that we used before, for a different population (we can compute the matrix once for all). In Identification Open Set the probe to identify might not belong to a subject included in the gallery, that's why we have a reject option, detection and identification rate at rank 1(that means for how many probes that belong to enrolled identities we get the correct detection, that is the similarity is below the distance threshold, and the correct identity is the first one in the list of those that fall below the distance threshold).
All gallery templates are matched against each probe. In a simplified version, each row is a single operation.
In this case it important who is in the gallery and who is not, just because we don't have a claimed identity. So we must evaluate the False Acceptance: FA only happens when we only detect a probe that is not in the gallery.
Each row is labeled with the probe ground truth identity only (but there is no claim).
Genuine probes are those belonging to identities in the gallery, impostor probes are those belonging to identities not in the gallery (not enrolled person).
Performance evaluation can be obtained by clearly separating probe and gallery (different subsets of templates so we must not have overlap between probe and gallery samples) and genuine (enrolled person) and impostors (identities that play the role of impostors are not included in the gallery, so they are not enrolled).

Example:

Identities A, B, C, D, E, F are in the dataset

Identities A, B, C, D are in the gallery with a single instance, so one template per identity

Identities E and F play the role of impostors just because they are not enrolled in the gallery

d() = function that measures distance form the probe

t = acceptance threshold

In this simple example, each row is a single identification operation and we only know the ground truth.

	Probes	A1	B1	C1	D1
A	P1	1	4	2	3
D	P2	4	1	3	2
E - Impostor	P3	4	2	1	3
C	P4	…	…	…	…
F - Impostor	P5	…	…	…	…

We have: P1, whose true identity is A: P2, whose true identity is D; P3, whose identity is not enrolled in the gallery.
The ordered list of distances for P1 is the following: d(A1), d(C1), d(D1), d(B1).
We may consider in general that whenever the distance from the correct identity is lower than the threshold, we can increase the number of the correct detect and identify outcomes at rank 1 for threshold t. In this case we don't only consider the threshold but also the fact that the template denoted by the correct identity is also the first one in the ordered list. In order then to compute the rate we must divide it for the number of probes that belong to people in the gallery. When such distance is above the threshold, we've False Rejection (that is always computed as the complement of Detection and Identification Rate at rank 1).
P2 is an enrolled person, so it should be recognized, because the identity D is enrolled in the gallery. The ordered list of distances for P2 is the following: d(B1), d(D1), d(C1), d(A1). The lower distance is provided by B1 so even if the distance is below the threshold we will have a False Rejection anyway because the distance with D1, that corresponds to the correct identity, is not in the first place in the ordered list. Even if the distance provided by D1 may be below the threshold and so we may have a correct detection, in this case we don't have a correct identification because there is a template with a lower distance that is wrongly returned as the first one.
The ordered list doesn't change if we change the threshold; we can only observe how the threshold is position within the list. So, in any case, with this distribution of values, we will never haver a correct identification for P2, because in any case we will always have B1 in the first position because it's more similar to the probe.
A possible outcome is that if the distance with the correct identity, that is D, is below the threshold that outcome will contribute to the count of Detection and Identification at rank 2 for threshold t. This is checked only if the first identity is lower than the threshold, because since we've an increasing list of identity if the first already higher than the threshold, obviously also the others will be higher respect the threshold.
If we have the distance of B1 higher than the threshold, in any case we have a False Rejection, because in any case not only P2 was not correct identified but also it was not even detected correctly.
In the case of P3, we've a not enrolled person, so we've an impostor. The ordered list of distances for P3 is the following: d(C1), d(B1), d(D1), d(A1). In this case we only have two possibilities: either a False Accept or a Genuine Reject. False Accept occurs whenever the first distance in the list is lower than or equal to the threshold, because since E doesn't belong to a subject that is enrolled in the gallery we will surely have a False Acceptance. Otherwise we've a Genuine Reject.

Example:

Identities A, B, C, D, E, F are in the dataset
Identities A, B, C, D are in the gallery with multiple instances
Identities E and F play the role of impostors just because they are not enrolled in the gallery
d() = function that measures distance form the probe
t = acceptance threshold

Also in this example, each row corresponds to a single identification operation.

	Probes	A1	A2	B1	B2	C1	C2	D1	D2
A	P1	2	1	8	4	3	7	5	6
D	P2	7	6	2	1	5	8	3	4
E - Impostor	P3	7	5	2	8	6	1	3	4
C	P4	…	…	…	…	…	…	…	…
F - Impostor	P5	…	…	…	…	…	…	…	…

We start from P1. The ordered list of distances for P1 is the following: d(A2), d(A1), d(C1), d(B2), d(D1), d(D2), d(C2), d(B1). In this case we can also choose to have in the returned list only the best outcomes for each identity. If the distance from A2 is lower than or equal to the threshold then we have a correct Detection and Identification Rate.
In this case we can have two possible outcomes: A1 and A2. If we' had only A1, we should have increase the false rejection if the threshold was set at 1. But since we also have another template with distance 1 in any case we increase the possibility to get the correct recognition.
We pass now to P2. The ordered list of distances for P2 is the following: d(B2), d(B1), d(D1), d(D2), d(C1), d(A2), d(A1), d(C2).
The correct identity for P2 is D, but we have B1 that is more similar to P2. So in any case we will never have a correct result for P2. We can only move the threshold to see what happens with the respect to detection and identification rate at ranks higher than 1.
So we have a False Rejection if the distance in the fist position in the list falls below the threshold, because it's not the correct one. If the it's above the threshold, in this case we consider both as a False Rejection.
The only positive outcome is that if the distance from the better D template that is D1 is lower than the threshold then we can increment Detection and Identification Rate at rank 3 for threshold t.
P3 is an impostor and its ordered list of distance is the following: d(C2), d(B1), d(D1), d(D2), d(A2), d(C1), d(A1), d(B2).
If the better value (so the first in the list) is lower than the threshold, we will have a False Acceptance. Otherwise we will have a Genuine Rejection.

We can carry out different evaluations by choosing probe set and gallery set in different ways. Each time it's possibly to have different distributions of enrolled probes and not enrolled probes, these distributions can influence the outcomes due to the effect of specific behaviors or specific subjects.
This approach requires to carry out a sufficient number of evaluations in order to compute a reliable average result:

- each time probe set and gallery sets are chosen in a different way
- each time there is a possibly different distribution of genuine probes (the identity of the probe is also in the gallery) and impostor probes (the identity of the probe hasn't been included in the gallery), i.e., how many genuine and how many impostors are considered.

Identification Closed Set is the easiest to evaluate because probe candidates always belong to subjects in the gallery; so there is no distinction between genuine and impostor/ enrolled and not enrolled.
All gallery templates are matched against the probe. Each rows is labeled with the probe ground truth identity (again we have no claim).
No impostor appears in the experiments (all identities are in the gallery). There is no acceptance threshold, so even the outcomes are simple to identify.
Performance evaluation can be obtained by clearly separating probe and gallery (different subsets of templates). So there are no overlaps between gallery samples and probe samples.

Example:

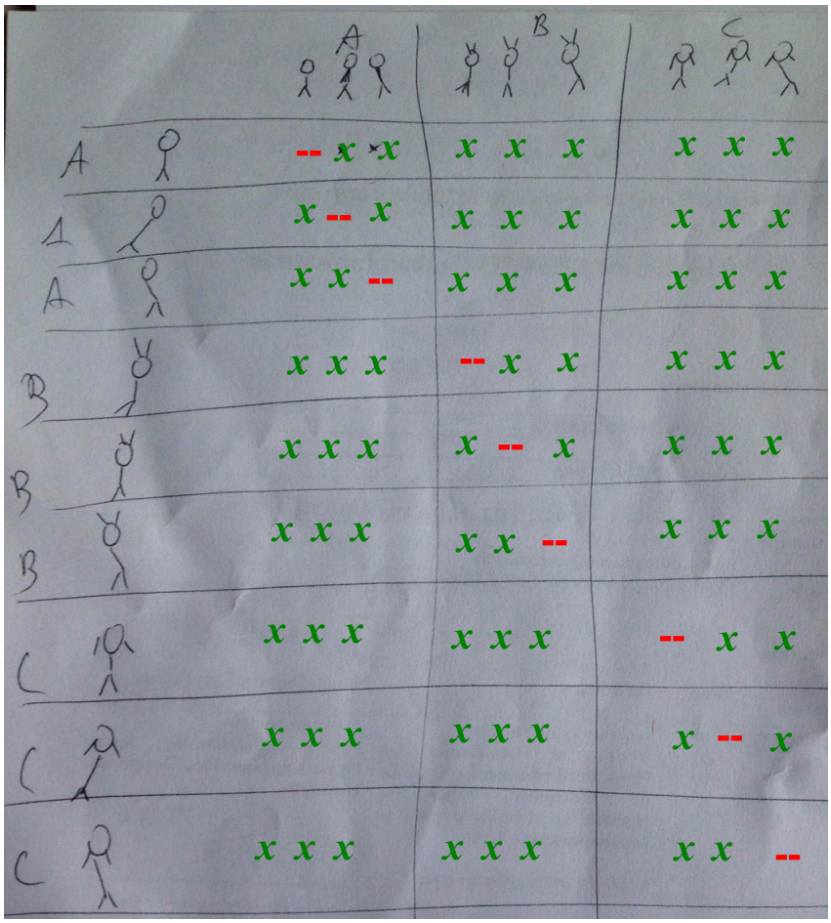
Identities A, B, C, D, E, F are in the dataset
Identities A, B, C, D, E, F are in the gallery with a single instance
d() = distance from the probe

Each row is a single identification operation. There isn't threshold.

	Probes	A1	B1	C1	D1	E1	F1
A	P1	1	4	2	3	6	5
D	P2	6	1	4	2	3	5
E	P3	5	2	1	3	4	6
C	P4	…	…	…	…		
F	P5	…	…	…	…		

We start from P1. For P1 the correct identity is A and we've the following ordered list of distances for P1: d(A1), d(C1), d(D1), d(B1), d(F1), d(E1).
The template of A is the correctly in the first position. So the identity in the first place is the right one. This result contributes to Cumulative Match Score (CMS) at rank 1, that we also called Recognition Rate (RR).
If we go to P2, its correct identity is D, but we've an ordered list where the first value is given by the distance from B: d(B1), d(D1), d(E1), d(C1), d(F1), d(A1). The identity in the first place is not the right one, but in the second place. So this result contributes to CMS at rank 2 and over (this because CMS aggregates all the ranks below k). So this outcome not only links to the CMS at rank 2, but also all the following values.
The identity of P3 is E and its ordered list is the following: d(C1), d(B1), d(D1), d(E1), d(A1), d(F1). The identity in the first place is not the right one, but it's in the fourth place. So this result contributes to CMS at rank 4 and over.
This approach requires to carry out a sufficient number of evaluations in order to compute a reliable average result: each time probe set and gallery sets are chosen in a different way.

Now we're going to consider the case when we're not going only to compute the matrix (exactly in the same way before), but also the case when we are going to use it in a massive way, in the sense that we're not going only to compute the complete All-against-All matrix but we are going also to consider each row as a different set of experiments. Different in the sense that each row represents more than one experiment.
This holds both for genuine impostors and for enrolled/not enrolled according to the kind of experiments that we want to carry out.
Also in this case distances are computed once and for all, and they are used in a different way according to the setup (modality and number of gallery templates per subject).
Each template plays in turn the role of either probe or gallery, possibly more than once.
Of course, diagonal values (comparisons of a a template with itself) are not considered.
Cumulative averages (rates) that are computed encompass many possible partitions.



In this first strategy in which we compute All-against-All, each template can be matched with all the others except itself. We can't never match a template against it self.
The distance/similarity matrix can be computed once and for all in advance. Then, according to the code that we're gong to apply for the evaluation, we can use it for different kinds of applications.
Each probe (row) is consider either as genuine or impostor in turn (if we're carrying out verification). This means that according to the number of identities in the gallery, a probe will play once the role of the genuine, and all the other times it will be an impostor. This increases the number of impostors in a verification system, so that we can farther stress the system.
The way that we get the results also depends of using single template strategies or multiple template strategies. The important thing that we must take into account is that each row can represent more than one experiment.
Results are accumulated to get the final statistics. Each row possibly. contributes more times according to the number of experiments it can possibly represent.
For simplicity, it is possible to assume the same number of samples per subject. But we can also have different number of samples per subject.

Which are the advantages to use this kind of strategy?
It's quite easy to program: once we've computed the overall matrix, we will compute a kind of «average» over all possible specific distributions of genuine/impostor attempts. The number of impostors is always much higher than genuine, therefore **it is possible to over-stress the system** to assess situations with many impostor attempts.

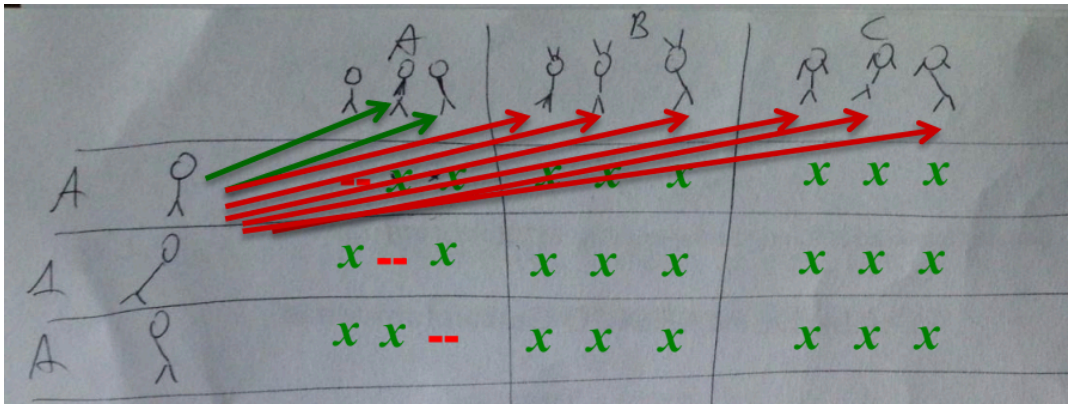
Which are the disadvantages?
The time requested to compute the full matrix may become very high.
It's not possible to analyze specific distributions of genuine/impostor attempts, that may achieve peculiar results.
This kind of computation is not suited when we have a dataset that has been acquired during different capture session separated in time. Samples that are acquired in the same session will be more similar to each other than those acquired in different sessions. If we mix them together the result could be better in a real situation.
Each row is either a single test operation or a set of tests, depending on the recognition modality.
We will further distinguish between single-template and multiple-template settings, depending on the number of samples per subject that are assumed to be stored in the test gallery. Once we have the overall matrix, it will be not difficult by software to partitioning it according to the strategy that we have chosen.
We assume that N is number of subjects; |G| is the cardinality of gallery that is the total number of samples (in this case it's also the number of rows and columns in the matrix); then we've S that is the number of templates per subject (|G| = S*N); i represents the row index (probe template) and label(i) is the associated identity; j is the column index (gallery template) and label(j) is the associated identity.
We're assuming to have a function label() that applied to a template returns the ground truth identity of that template.

VERIFICATION SINGLE-TEMPLATE

When we want to use the distance matrix for verification single-template, we can assume that each row is a set of |G|-1 operations (there is a '-1' because we must always discard the comparison of the template with it self). Each time the probe declares a different identity.
How many times this identity is genuine? Each rows contains S-1 genuine attempts, that is all the template of the correct subject minus the same that is used as probe. On the other hand, **each row contains (N-1)*S impostors**. So the number of all the other subjects times the number of templates for that subject.
How many Total Genuine Attempts shall we have?
Remember that the total number of Genuine Attempts is used to compute FRR and GAR. We know that for each row we've S-1 genuine attempts and we also know that the number of rows is equal to the number of samples |G|; so the number of Total Genuine Attempts is given by: **TG = |G|*(S-1)**.
How many Total Impostors Attempts shall we have?
For each row we have (N-1)*S impostor attempts, so the number of Total Impostors Attempts is given by: **TI = |G|*(N-1)*S**.

```
for each threshold t
    for each cell M_{i,j} with i≠ j
        if M_{i,j} ≤ t then
            if label(i)=label(j) then GA++
            else FA++
        else if label(i)=label(j) then FR++
                else GR++
    GAR(t)=GA/TG; FAR(t)=FA/TI;
    FRR(t)=FR/TG; GRR(t)=GR/TI
```

For each threshold we must compute the statistics of errors and then we can draw the curves.
For each threshold we analyze in turn each cell of the matrix except for the cells that lie down the diagonal.
We're considering each single cell in turn; but in turn we're considering that single cell according to the label because we consider the column index as the claim and the row index as the correct identity. This is an implicit claim given by the column that we're considering.
So given a certain cell $M_{i,j}$ we first check if the value is lower than or equal to the distance threshold. Once we've done the comparison with the threshold, we check the identities of the row and of the column. If the label of the row and the label of the column are the same, this means that the probe has a genuine claim. Since $M_{i,j}$ was below the distance threshold, then we can increase the number of Genuine Acceptance. Otherwise, the labels are different (so we've a false claim) but $M_{i,j}$ was below the threshold, so we can increase the number of False Acceptance.
If we have a cell $M_{i,j}$ whose value is higher than the threshold, we check again the labels and if they are the same it means the it was a genuine claim; in this case we can increase the number of False Rejection. If the labels are different and the value is above the acceptance threshold, we can increase the number of Genuine Rejection.
Once we've inspected the overall matrix, we just to divide the aggregated values with that we've calculated before. The Genuine Acceptance Rate is given by the division between the number of Genuine Acceptance by the number of Genuine Claims:
GAR = GA/TG.
The False Acceptance Rate is obtained by the number of the False Acceptance divided by the total number of impostors: **FAR = FA/TI**.
And the same for FRR and GRR: **FRR = FR/TG** and **GRR = GR/TI**.

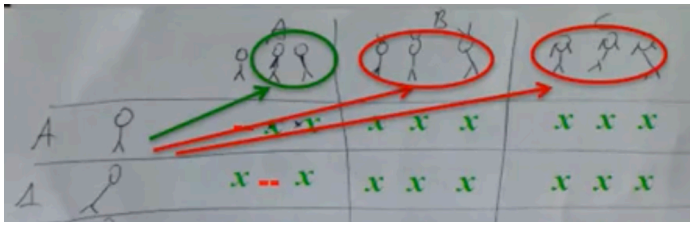


VERIFICATION MULTIPLE-TEMPLATE

Exactly with the same matrix, we're taking into account Verification Multiple-Template. Again each row doesn't represent a single experiment but more than one, but a lower number of experiments respect before. This because this time we're not going to compare the probe template by template but with group of template (a common strategy for example is to take the best result given a certain claimed identity). So in this case each row is a set of N operations, because each time that the probe claims a certain identity it is compared against all the template for that identity. In other words, for each row we have any group of comparisons; one of which is genuine and N-1 are impostors attempts.
So the number of Total Genuine Attempts is given by |G|, while the number of Total Impostors Attempts is given by |G|*(N-1).

```
for each threshold t
    for each row i
        for each group M_label of cells M_i,j with same label(j)
            excluding M_i,i
            select  diff = min(M_label )
            if diff ≤ t then
                if label(i)=label(M_label) then GA++
                else FA++
            else if label(i)=label(M_label) then FR++
                else GR++
    GAR(t)=GA/TG; FAR(t)=FA/TI;
    FRR(t)=FR/TG; GRR(t)=GR/TI
```

The pseudocode is similar to the previous one; for each threshold we do an inspection of the matrix. For each row we consider a group of cells $M_{i,j}$ that have the same label(j); this simply means that we consider a group of templates for the same identity. When two rows have the same label, it means that they corresponds to the same identity.
We exclude again the diagonal in any kind of computation and we may select a value for each group of templates corresponding to the same identity (for example, the minimum value in that group).
If such minimum distance is lower than the threshold then we must check whether the label of the probe is the same as the label of this group of templates and then we increase the number of Genuine Acceptance.
Otherwise if the distance is lower than the threshold but the labels are different, we increase the False Acceptance.
Else if the distance is above the threshold, if the label of the probe and the label of the group of templates are the same, we increase the number of False Rejection. Otherwise it is a Genuine Rejection.
At the end of the second for, we compute the statistic for that threshold:
GAR = GA/TG;
FAR = FA/TI;
FRR = FT/TG;
GRR = GR/TI.



IDENTIFICATION OPEN SET

In this case, if we consider templates one by one as for verification, the pseudocode would become complicated. So to avoid having a too complicated pseudocode, we only consider Identification open set multiple-template. But it's possible also to consider a scenario where each template corresponds to a single template in the gallery.
While in verification we've a one-shot comparison with the claimed identity, in this case, considering single templates in the gallery and considering that the operation returns an ordered list, we should consider all the possible combinations for templates. So all the possible returned lists.
We fall again in condition where a specific choice may influence the outcome.
Each row is a set of 2 identification operations. This means that each row (probe) can be considered either as an enrolled identity or not. So each row contains 1 genuine attempt (subject in the gallery) and 1 impostor attempt (subject not in the gallery). In this case we only have two experiments because since we don't have claim by the user, even if the user is in the gallery or not.
So the number of **Total Genuine Attempts** is given by the total number of samples: **TG = |G|**. Also the number of **Total Impostor Attempts** is given by the total number of samples, so: **TI = |G|**. Because each row contains both a genuine attempt and a impostor attempt.

for each threshold t
for each row i
$\{L_{i,m} | m=1 \dots |G|-1\} =$
$\{M_{i,j} | j=1, \dots |G|\} \setminus M_{i,i}$ ordered by increasing value (the identical element is excluded)

if $L_{i,1} \leq t$ then (potential accept)
if label(i)=label($L_{i,1}$) then DI(t, 1)++ (genuine case)

find the first $L_{i,k}$ such that label($L_{i,k}$)<>label(i) AND $L_{i,k} \leq t$
if this k exists, then FA++ (impostor case: jump the label)

else find the first $L_{i,k}$ such that (if genuine yet not the first, look for higher ranks)
label(i)=label($L_{i,k}$) AND $L_{i,k} \leq t$
if this k exists, then DI(t, k)++ (end of genuine)
FA++ (impostor in parallel, distance below t but different label)
(no need to jump since the first label is not the «impostor»)

else GR++ (impostor case counted directly, FR computed through DIR)

DIR(t,1)=DI/TG; FRR(t)= 1- DIR(t,1)
FAR(t)=FA/TI; GRR(t)=GR/TI

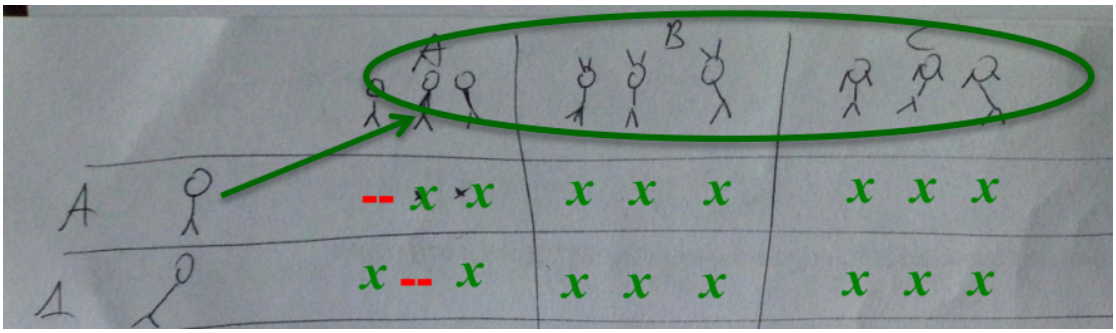
k=2 (higher ranks)
while DI(t, k)≠0
DIR(t, k)=DI(t, k)/TG+DIR(t, k-1) (we have to compute rates)

We're taking into account for each row the two cases when the subject belongs to the gallery or not.
In this case we've to consider that the distance values of the same row must be ordered in increasing order in order to get the candidates list that would be returned by an Identification Open Set operation.
So we consider $L_{i,m}$ as the ordered list of distances from the i row of the matrix (except for the diagonal $M_{i,i}$).
The first item that we check is the first item in list $L_{i,1}$. We consider first if this element is lower than or equal to the distance threshold. We can now consider in parallel different cases: if the label of the probe and the label of the first element in the list are the same, we can increase the number of Detection and Identification at rank 1 for threshold t; at the same time we can also increase the number of False Acceptance because there is the case when the probe doesn't belong to the a person in the gallery.
In the same condition we can also find the first $L_{i,k}$ such that the label of $L_{i,k}$ is different from the label of the probe and $L_{i,k}$ is lower than the threshold. If this k exists, then we can increase the number of False Acceptance.
We're still in the condition in which the first element in the list is lower than the threshold: so if the label of the first element is the same as the probe, we can increase DI(t,1) and this is the case where the person is genuine; in parallel, we're considering the case where the person is not enrolled.
We cannot rely on the first place of the list even if it is equal to the identity of i, because if the first place in the list is equal to the identity of i and the identity of i is not in the gallery, we would not have that value in the head of the list in a real operation.
So, in order to consider in parallel the case when the subject is not enrolled in the gallery, we don't consider the first case if it has the same label as the probe, but we go to look for the first position in the list with a label different from the probe. Because we're considering the opposite case when the probe is not in the gallery.
If this is lower than the threshold, this means that even though the identity of probe i is not in the gallery, there is a template that compared with the probe i provides a value lower than the threshold. If this template exists, we can also increment the False Accept.
If $L_{i,1}$ is lower than the threshold but the top label is different from the label(i), we go to find the first $L_{i,k}$ such that it has the same label as the probe i and an acceptable distance lower than the threshold. If this k exists, we can increment the number of Detection and Identification at rank 1 for the threshold t.
At the same time, we can increase the number of False Acceptance because we're in the case where the first element in the list has a distance that is lower than the threshold and its label is different from the label of the probe.
Otherwise if $L_{i,1}$ is higher than the threshold we can increase the Genuine Rejection because we're in the case that the distance is over the threshold so we're considering the case where the probe is not in the gallery. The impostor case can be counted directly.
Then we compute Detection and Identification Rate by just divides the number of Detection and identification by the number of Genuine Attempts: **DIR(t,1) = DI/TG**.
False Rejection Rate is just 1-DIR at rank 1: **FRR(t) = 1 - DIR(t, 1)**.
False Acceptance Rate: **FAR(t) = FA/TI**.
Genuine Rejection Rate: **GRR(t) = GR/TI**.
The only difference here it that we also compute the increasing Detection and Identification at ranks higher than 1. This because we've to accumulate the values below k in order to get DIR at rank k.

IDENTIFICATION CLOSED SET

In the case of Identification Closed Set the computation is simpler because we don't have any threshold, we assume that every probe belongs to the a subject in the gallery and we only take into account the rank where the correct identity is returned in order to evaluate the overall behavior of the system. Even though the system is not able to retrieve the correct identity in the first case, maybe it is useful to know if the system is capable to retrieve the correct identity within x-th place.

for each row i
  $\{L_{i,m} | m=1 \dots |G|-1\} =$
  $\{M_{i,j} | j=1, \dots |G|\} \setminus M_{i,i}$ ordered by increasing value
  find the first $L_{i,k}$ such that
    label(i)=label($L_{i,k}$)
  CMS(k)++
CMS(1)=CMS(1)/TA; RR=CMS(1)
k=2
while k< |G|-1
  CMS(k)=CMS(k)/TA+CMS(k-1)



All-against-All is not suited for specific situations where we have precise partitions of the dataset in two different sessions.
When there is a precise subdivision of samples into sessions captured during non-overlapping sessions at different times, it is more fair to create probe and gallery sets using templates from different sessions.
This because it happens that samples that are captured in in the same session are usually more similar than those from different sessions, due to variations happening with time (e.g., slight face variations)/environmental conditions (e.g. illumination)/sensor performance (e.g., dirt deposited on a fingerprint sensor). Therefore the overall performance may appear better.
It is suited for situations where we do not have a known/relevant time lapse among the different templates of a same subject.
What can we do when we have well defined sessions in a dataset, that means group of samples that have been annotated with a different kind of capture?
We can use the schema Probe vs Gallery, but in a more efficient way.

ALL-AGAINST-ALL PROBE VS GALLERY

In this case, instead of choosing a subset of probe and the gallery, we can use an entire session as gallery and an entire session as probe (this is the easiest example: when we have only two sessions; but this can be also repeated when we have more than two sessions, such that we have a subset of sessions as probe and subset of sessions as gallery).
This can also be useful when we must partition our dataset taking into account the need for training: one session can be used for training only.
For session we mean a group of samples that are acquired at the same time for different subject.
Since we use a session for the probe and the other for the gallery in this case, we don't choose a specific subsets of probes but we take all the session as the probe.
We can apply the same reason that we apply for the overall All-against-All distance matrix. So, also in this case, we can compute the overall distance matrix in advance: all probe templates are compared with all gallery templates. But this time we cannot have a probe that can be in turn either in the probe and in the gallery.
Also in this case distances are computed once and for all, and they are used in a different way according to the setup (modality and number of gallery templates per subject).
Each template plays in turn the role of either genuine/impostor or enrolled/not enrolled more than once according to the recognition application (see the different cases).
It could be useful, once we've completed the computation for one configuration, to create different probe/gallery partitions and average the results. For example we can swap the role of the sessions: the session that it was used for the gallery becomes the probe and vice versa and then to average the result that we've achieved.
Each probe, that is each row, is considered either as genuine/enrolled or impostor/not enrolled in turn.
The results are accumulated to get the final statistics.
Even in this case, each row possibly contributes more times according to the number of experiments it can possibly represent. The number of experiments that a probe can represent is more or less the same that we've seen before.
Again, for simplicity, it is possible to assume the same number of samples per subject.
We've also more or less the same pros and cons that we had in the previous schema.

Which are the advantages in this case?

It's easy to program, in practice computes a kind of «average» over all possible specific distributions of genuine/impostor (enrolled/not enrolled) attempts. Even in this case, the number of impostors is still much higher than genuine, therefore it is possible to over-stress the system to assess situations with many impostor attempts. Because whatever is the strategy that we adopted, in any case the number of impostors will be higher than the genuine.

Which are the disadvantages?

Even in this case we have a possibly high computational time and we don't analyze specific distributions of genuine/impostor /enrolled/not enrolled) attempts, that may achieve peculiar results.
This kind of computation is possible only when we have a ground truth.
Also in this case, each row is either a single test operation or a set of tests, depending on the recognition modality.
We will again distinguish between single-template and multiple-template settings, depending on the number of samples per subject that are assumed to be stored in the test gallery.

N is the total number of subjects (all contributing to both probe and gallery). This is not always true: especially in some dataset when there is a specific distinction between the sessions, we may find that there is a subset of subjects that only participates in one session. This subset is a good candidate either to be used in the testing or to be used as probe, because they represented person that are not enrolled in the system.
Then we have **|G|** that is the **total number of gallery samples** (that is the number of columns in the matrix) ; **|P|** is the **total number of probe samples** (that is the number of rows in the matrix).
This time, no samples are in common between probe and gallery.
For simplicity 2S is the number of templates per subject, so that, for simplify the code, we can consider that both |G| and |P| are equal to S*N: **|G| = |P| = S*N**.
Again we have i that is the probe template index (row index) and a ground truth function label(i) that returns the correct identity of that template. In the same way, we've j that is the gallery template index (column index) with the same function label(j) that returns the correct identity.

VERIFICATION SINGLE TEMPLATE

```
for each threshold t
    for each cell M_{i,j}
        if M_{i,j} ≤ t then
            if label(i)=label(j) then GA++
            else FA++
        else if label(i)=label(j) then FR++
                else GR++
    GAR(t)=GA/TG; FAR(t)=FA/TI;
    FRR(t)=FR/TG; GRR(t)=GR/TI
```

In this case we've the same kind of possibilities that we had we the pure All-against-All matrix, in the sense that each row is a set of exactly |G| operations, because in this case we've S genuine attempts for the same probe (this happens everytime this person claims the true identity and in turn we check one of S templates for that person, assuming that the gallery only contains one template per person).
At the same time, each row contains (N-1)*S impostors attempts. That is everytime this probe claims a different identity then the true one, it does it S times for each identity (this because the gallery contains in this case only one template per identity). In order to compare the rates, it's necessary to compute the **Total Genuine Attempts** and the **Total Impostor Attempts**:
TG = |P| * S
TI = |P| * (N-1) * S
The difference between this code and that one that we used for the pure All-against-All is that here don't have anymore the check i different from j, because we don't have anymore the concept of diagonal, because the same template will never appear in the same row and column.
If we have the value in the call that is lower the threshold, we check if the labels associated to the row and column are the same; if they are, we have a Genuine Acceptance, otherwise we've a False Acceptance.
If the value in the cell is above the threshold, we check again the labels of the two templates: if they are the same, we've a False Rejection, otherwise we've a Genuine Rejection.
Once we've completed the cycle for all the cells, we have to compute:
GAR = GA/TG
FAR = FA/TI
GRR = GR/TI
FRR = FR/TG

VERIFICATION MULTIPLE-TEMPLATE

for each threshold t
  for each row i
    for each group M_{label} of cells $M_{i,j}$ with same label(j)
      select diff = min(M_{label})
      if diff $\leq$ t then
        if label(i)=label(M_{label}) then GA++
        else FA++
      else if label(i)=label(M_{label}) then FR++
        else GR++
  GAR(t)=GA/TG; FAR(t)=FA/TI;
  FRR(t)=FR/TG; GRR(t)=GR/TI

In this case we take for each row and so for each identity claimed, we take the overall group of templates with the same label. We assume that in turn row/ probe i declares a different identity: one time this identity is genuine, while all the others N-1 times this identity is false.
So each row contains 1 genuine attempt and N-1 impostor attempts.
This time, the **Total Genuine Attempts is equal to |P| and the Total Impostor Attempts is |P| * (N-1)**.
We don't multiply for S because each group of templates works for a single identity verification; we look at all the distances in that group, we select the minimum and then we go ahead by considering the relationship of this value with the threshold and we consider the possible equality of the two labels.
We may also take the average of the distances in the group instead of taking the minimum.

IDENTIFICATION OPEN SET

for each threshold t
  for each row i
    $\{L_{i,m} | m=1 \dots |G|\}$ =
    $\{M_{i,j} | j=1, \dots |G|\}$ ordered by increasing value

    if $L_{i,1} \leq$ t then (potential accept)
      if label(i)=label($L_{i,1}$) then DI(t, 1)++ (label(i) enrolled)

        find the first $L_{i,k}$ such that label($L_{i,k}$)<>label(i) AND $L_{i,k} \leq t$
        if this k exists, then FA++ (label(i) not enrolled, jump it)

      else find the first $L_{i,k}$ such that (if genuine yet not the first, look for higher ranks)
        label(i)=label($L_{i,k}$) AND $L_{i,k} \leq t$
        if this k exists, then DI(t, k)++ (end of genuine)
        FA++ (impostor in parallel, distance below t but different label)
        (no need to jump since the first label is not the «impostor»)

      else GR++ (impostor case counted directly, FR computed through DIR)

  DIR(t,1)=DI/TG; FRR(t)= 1- DIR(t,1)
  FAR(t)=FA/TI; GRR(t)=GR/TI

  k=2 (higher ranks)
  while DI(t, k)≠0
    DIR(t, k)=DI(t, k)/TG+DIR(t, k-1) (we have to compute rates)

In this case, each row is a set of 2 identification operations, in the sense that one time we assume that the person is in the gallery, the other time we assume that the person is not in the gallery.

So, each row contains 1 «genuine» attempt (subject enrolled in the gallery) and 1 «impostor» attempt (subject not enrolled in the gallery).
The Total Genuine Attempts and theTotal Impostors Attempts have the the same cardinality of the probe set |P|.
Also in this case, we don't have to check if the cell doesn't correspond to the comparison of the template with it self.
Notice the computation of DIR at ranks higher than k: here is evident how this computation entails aggregating DIR at lower ranks when we want to compute the DIR at a certain rank.

IDENTIFICATION CLOSED SET

for each row i
{$L_{i,m}$| m=1 … |G|} =
{$M_{i,j}$ |j=1, …|G|} ordered by increasing value
find the first $L_{i,k}$ such that
label(i)=label($L_{i,k}$)
CMS(k)++
CMS(1)=CMS(1)/TA; RR=CMS(1)
k=2
while k< |G|
CMS(k)=CMS(k)/TA+CMS(k-1)

In this case, each row is an operation and contains 1 genuine attempt. So the Total Attempts is equal to |P|. We assume that there are not impostor attempts and also we don't use acceptance threshold.

Why is it so important to carry out the correct calculation, especially when rates are involved?
Let us consider the following example, related to verification mode. We have 100 probes.
Let us assume that the system erroneously accepts 10 impostors, and erroneously rejects 10 genuine users.
So we only know the absolutely numbers of errors.
If we compute FAR and FRR with respect to the total number of probles, we would get FAR=FRR=10/100. So in this case we would be saying that FAR and FRR are the same.
THIS IS WRONG.
If we compute the FAR and the FRR with the respect to the total number of probes we obtained in both cases 0.1 error rate. But something missing.
Let us further assume that the impostor probes are actually 10, and genuine probes are actually 90.
According to the previous computation, we would still get FRR=FAR=10/100.
We must consider that over 100 probes 10 are impostors and 90 genuine. So, in this case, all the impostors were accepted, and only 1 each 9 genuine users was rejected.

Probes	Genuine	Impostors	FR	FA	FRR	FAR
100	90	10	10	10	10/90=0.11	10/10=1 (!!!)
100	50	50	10	10	10/50=0.2	10/50=0.2
100	10	90	10	10	10/10=1 (!!!)	10/90=0.11

In practice, the identification task is much more difficult for biometric systems (but also for human operators) than the verification task, because we must both meet the acceptance threshold and return the correct person as the recognized identity.
So that it's important to take into account the different kind of tasks to exploit the correct statistics.
The performance statistics are different for different kinds of application just for this reason.
It is critical to think in terms of the proper task, and their associated statistics, in order to avoid confusion and errors.
Which are the possible solutions not much to measure the reliability but rather solve the problem of detecting a low reliability of responses especially when considering a specific individual?
One possible solution to increase the reliability of the system is not only to have more templates but also to **update the templates in the gallery**. Once we've a gallery of stored templates, when we're sure enough about the recognition operation, we may decided to include new probe within the gallery. Possibly delating some less informative templates.
If we also maintain the old templates, we only increase the sub-gallery for a certain subject, but this also can provide better solution intra-class variations.
Template updating becomes mandatory in the case of edging of the subjects for certain biometric traits.
Another solution is to **address the problem of poor quality templates**. For example, if we have a fingerprint that has been enrolled for some reasons with a low resolution sensors, once a new sensor is buyed, as new probes arrived, we include probes captured with the new sensor in the updated gallery.
Either we rely on the fact that we know that the identity was correctly recognized otherwise we may apply some other kind of solutions for template update. Because we may have either a **Supervised System**, so there is a person (a supervisor) that once a person is recognized determines if that recognition was correct and decides to include template in the gallery.
Otherwise we may have a **Semi-Supervised approach** so that it may happen in automatic way (there is no person supervising the process). Then we may try to apply some kind of statistics to compare the new result with the existing gallery.
We can also **select the most representative templates** (after adding much more templates, the identification becomes too huge). The selection of the most most representative templates to use for updating can be carried out **Online** (the selection is performed as soon as new input data is acquired by the recognition system) or **Offline** (the selection is performed after a certain amount of data has been acquired during a specific time lapse).

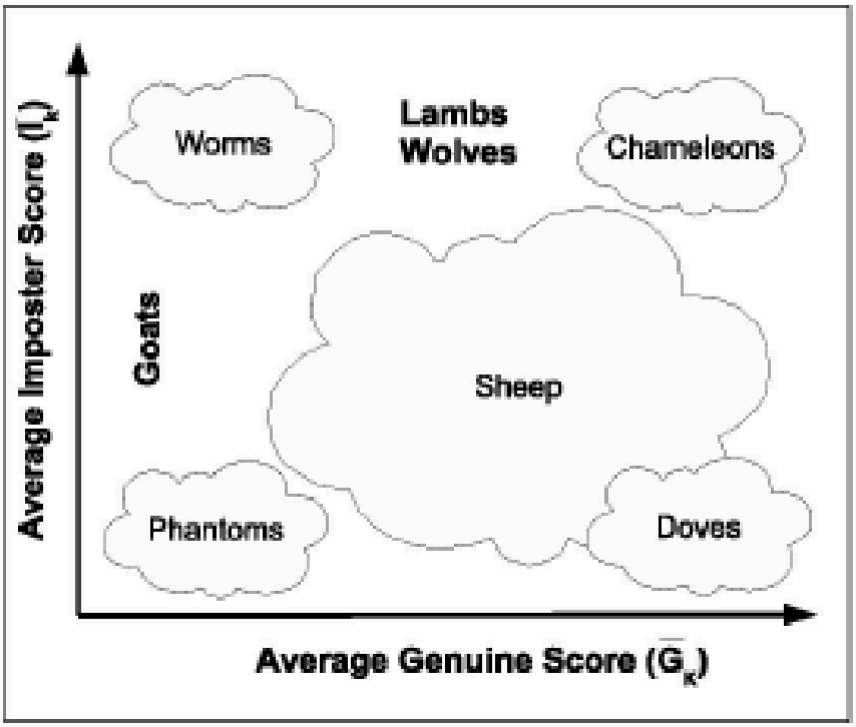
DODDINGTON ZOO

Most errors in the system can be attributed to specific class of users. The fact that a person belongs to one of the bad classes of Doddington zoo may affect the reliability of the single recognition operation.
Doddington defined Sheep, Goats, Lambs and Wolves in the context of speaker recognition systems:

- **Sheep**: A person who is a Sheep produces a biometric sample that matches well to other biometric samples of the same person and poorly to those of other people. As such, Sheep generate fewer false accepts and rejects than average (Normal average behaviour). This is the only good class because a person who is a Sheep will mostly achieve Genuine Acceptance and seldom achieve a False Acceptance when it claims a different identity.
- **Goats**: A person who is a Goat produces a biometric sample that poorly matches to the other biometric samples of itself. These low match scores imply a higher than average false reject rate for Goats. This is a person that achieves higher than average FRR because this person is not well recognized.
- **Lambs**: A person who is a Lamb can be easily impersonated. When the biometric sample of such a person is paired to a biometric sample from a different person, the resulting match score will be higher than average. Consequently, false matches are more likely. This may happen for example for children.
- **Wolves**: A person who is a Wolf is good at impersonation. When such a person presents a biometric sample for comparison they have an above average chance of generating a higher than average match score when compared to a stored biometric of a different person. So we have a lot of False Acceptances.

Goats, lambs, and wolves are defined in terms of a user's average genuine or imposter scores. If we consider both kind of errors, we may increase the number of classes.
New additions to the biometric menagerie by **Yager** and **Dunstone** are defined in terms of both a user's genuine and imposter scores.
Consider a user population P and a set of verification match score S. For each pair $j, k \in P$, there is a set of scores $\{s(j, k)\} \subset S$ containing all the verification results obtained by matching one template of j with the template of k.
User k's genuine scores is the set $Gk = \{s(k, k)\}$ and k's impostor scores is the set $Ik = \{s(j, k)\} \cup \{s(k, j)\}$ for all $j \neq k$.
Chameleons always appears similar to others. That is something slightly different from Wolves: in practice, Chameleons have high both average score with it compared with itself and high average score when it plays the role of impostor. So they receive high match score for all verifications. For this reason, Chameleons rarely cause False Rejects, but are likely to cause False Accepts.
An example of a user who may be a Chameleon is someone who has very generic features that are weighted heavily by the matching algorithm.
Phantoms lead to low match scores regardless of who they are being matched against. So Phantoms have a low average genuine score and also a low average impostor score. Phantoms may be the cause of False Rejections but unlikely to be involved in False Accepts.
A potential cause for Phantoms would be a group of people who have trouble enrolling in the system. This would lead to feature extraction difficulties and consequently low match scores for all verification. For example a person with a poor skin on the fingerprints can have problems with automatic enrolling.
Doves have the high genuine average score and lower average impostor scores, so this is something even better than Sheeps. They are pure and recognizable, matching well

against themselves and poorly against others. Doves are rarely involved in any type of verification error.
In a biometric system, Doves may be users who have an uncommon characteristic (for example a very distinctive nose). This would lead to both high genuine match scores and low impostor scores.
Worms are the worst conceivable users because they have a low genuine average score but a high average impostor score. They have few distinguish characteristics so few are correctly recognized as themselves but they can successfully impersonate other people.



In order to take into account in a more systematic way the different contributions of FA and FR to the performance of a system, it's also possible to consider a different measure that is **Decidability Value**. This value has been used in challenges of iris recognition.
Once we use a template, whatever is the composition of the template, we obtain a distance. We perform a one-against-all comparison: for each image in the dataset we compute a set of inter-class dissimilarity values, that is the distance value of this image with templates of the same subject.
So we collect in a set D^I the intra-class dissimilarity values and the inter-class dissimilarity values into another set that is D^E.
The Decidability Value is given by the difference in the average value in the set D^I and the set D^E divided by the standard deviation in such two sets in order to have a normalized measure.
The problem of reliability is a little bit different then considering the overall accuracy of a recognition system. Different kind of users with different trait characteristics can increase just for the testing operations related to those users either False Recognition or False Acceptance. We can also have samples with low quality, for example an over-illuminated face image or a fingerprint taken by a dry fingertips. So it may happen with a good system that from time to time some samples provide wrong results.
In general, in these cases, also we've to consider that measures like FAR, FRR, CMS, ... (all those performance measures that we've consider up to now) are not enough to give a thorough evaluation of algorithms.

First of all, because they are **Ex-post measures**; this means that we've anyway a ground truth then can guide our evaluation and in any case the benchmark, that is the dataset over which we carry out our testing, is extremely important for the generalizability of the result.
When we come to have a real operation, the operation context can be different or even change during time (for example a user that is not familiar with the system can arrive), so the score distributions may also change in turn.
For reliable comparison of systems we have to consider:
- the number and characteristics of the databases used;
- size of images (larger images may represent higher resolution);
- size of Probe and Gallery (in particular the relative size);
- amount and quality of addressed as well as tolerated variations;
- possible interoperability (e.g., cross-dataset generalization).

From **Torralba**: «Datasets are an integral part of contemporary object recognition research. They have been the chief reason for the considerable progress in the field, not just as source of large amounts of training data, but also as means of measuring and comparing performance of competing algorithms.» All the approaches was tested private datasets so that everybody was able to claim anything regarding performance. The big advance started when public available datasets started to be collected.
«At the same time, datasets have often been blamed for narrowing the focus of object recognition research,» because each dataset is usually collected under precise conditions.
«reducing it to a single benchmark performance number. Indeed, some datasets, that started out as data capture efforts aimed at representing the visual world, have become closed worlds unto themselves (e.g. the Corel world, the Caltech-101 world, the PASCAL VOC world). With the focus on beating the latest benchmark numbers on the latest dataset, have we perhaps lost sight of the original purpose?»
How can we deal with a distance matrix?
For each pair($probe_i$,$gallery_j$) we compute the distance and put it in the corresponding position of a distance matrix DM. In general such matrices are **symmetric with null diagonal**, in the sense that we assume that the distance measure that we're using is semimetric at least.

$$\begin{pmatrix} 0 & d_{12} & d_{13} & d_{14} \\ d_{12} & 0 & d_{23} & d_{24} \\ d_{13} & d_{23} & 0 & d_{34} \\ d_{14} & d_{24} & d_{34} & 0 \end{pmatrix}$$

Semimetric means that the distance between an element and itself is 0 and that if we have elements A and B, distance between A and B is the same as the distance between B and A.
Then, in order to have a matrix, we've to add the so **called triangular rule**.
This means that if we have three points A, B and C, the sum of the distances AB+BC is always lower than the distance AC: so if we've an intermediate point, the distance can only increase.
It is quite easy to transform a non symmetric measure into a symmetric one by taking the sum in the two directions and dividing by 2 (taking the average).
A **metric** on a set X is a **function** (called the distance function or simply distance).
This distance function takes a pair of objects from set X and returns a real number (**$d : X \times X \rightarrow R$**). This real number is the so called **distance**.

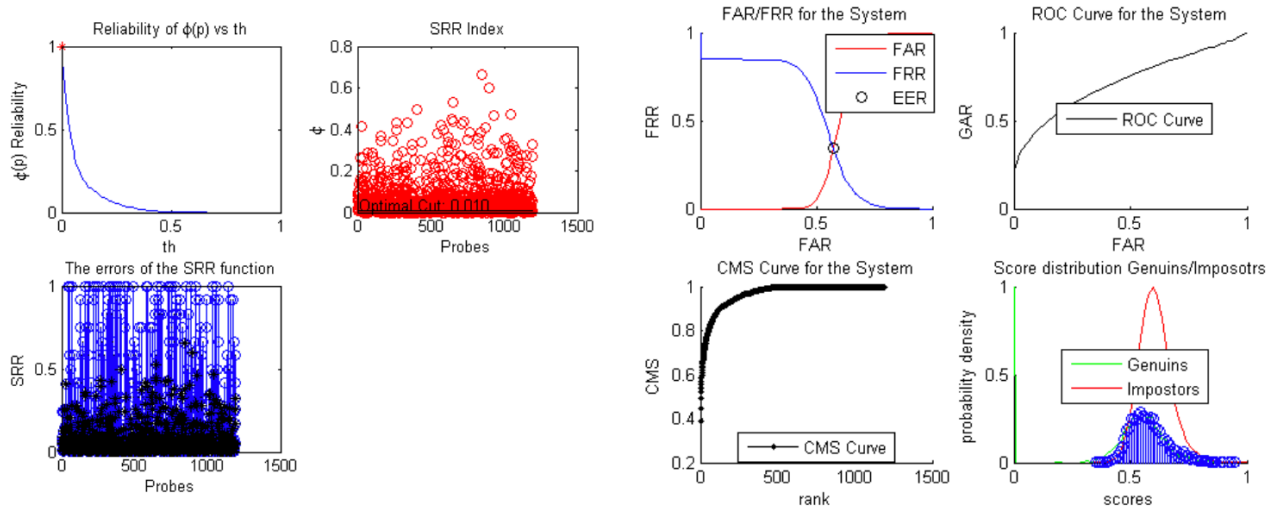
If we've three elements x, y, z in X, this function is required to satisfy the following conditions:

- d(x, y) ≥ 0 (non-negativity or **separation axiom**)
- d(x, y) = 0 if and only if x = y (identity of indiscernible or **coincidence axiom**)
- d(x, y) = d(y, x) (**symmetry**)
- d(x, z) ≤ d(x, y) + d(y, z) (**sub-additivity / triangle inequality**)

A **semimetric on X** is a function **d : X × X → R** that satisfies the first three axioms, but not necessarily the triangle inequality.

How can we compute performance measures from distance matrix?

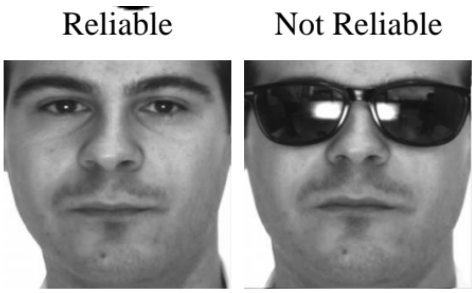
Starting from the distance matrix DM it is possible to compute many performance measures, often called **Figures of Merit (FoMs)**.



RELIABILITY OF AN IDENTIFICATION SYSTEM

Due to the possible different quality of input to different systems, and to possible accuracy in recognition procedures, it may happen that, notwithstanding global FOMs, not all responses are equally reliable.

For example, it may happen that we can have two face images of the same person, one with glasses and the other one without it; this suggest that since we miss an important element (the periocular region) to recognize a person, we may say that the recognition given by the image with glasses may be less reliable.



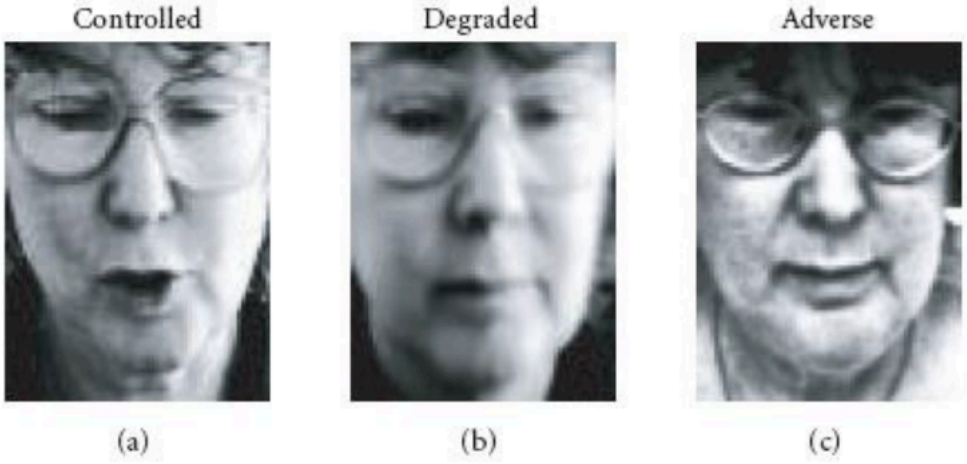
The definition of a reliability measure for each single response from a system provides further information to be used in setting up an operation policy (e.g., is f the identification is not reliable enough and if possible, repeat capture), but also to merge results from different systems (multibiometric architectures).

So the problem of reliability is not related to the overall accuracy of the system but to how we can trust the single decision by the system according to the samples that it have.

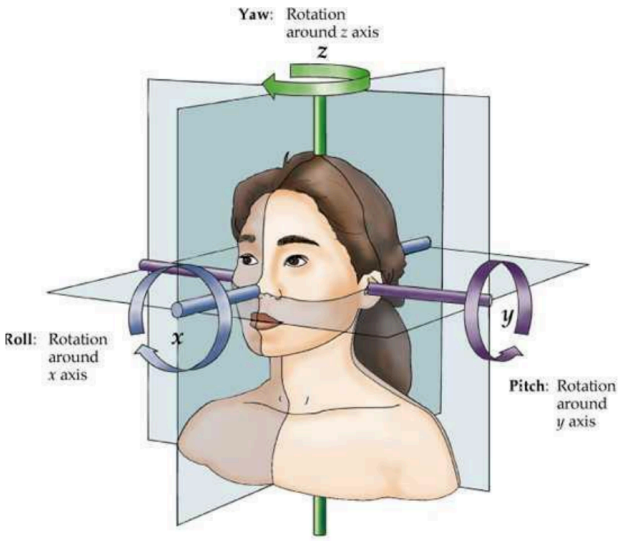
One possibility to check in advance the possible reliability of a recognition operation could be analyze the **quality of an incoming image**.

From the BANCA database for example, there are three subsets of images that are divided according to the characteristics of the image captured.

We have an example of Controlled capture, in the sense that we've a sharp image with a good contrast, with even illumination and so we can consider it as a good image (it's taken in controlled conditions).
Then we have the Degraded group of images, where we've some blur given by movement or by out of focus, but anyway we don't lose so many details.
In the last image, we've Adverse conditions: there is high contrast and a big shadow on the upper part of the image. This may make the recognition relying also in that part of image less effective.



We can measure the correlation with a so called "**average image**", in the sense that we can create an average template from all faces (regarding illuminations, sharpness, whatever), and then we take the quality of this as a reference.
In addition, we can **estimate the sharpness of images**, because the lack of high frequency details in the image can be described as a loss of sharpness (blurring).
Another way to measure the quality of face image is to consider the deviations of the pose.

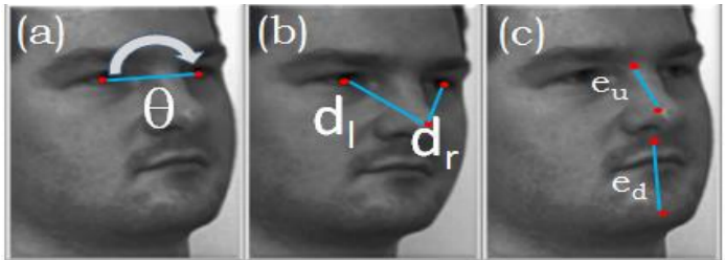


We have; the **Roll**, that is the rotation around the x axis (the x axis is the one perpendicular to the face); the **Pitch**, that is the rotation around the y axis; the **Yaw**, that is the rotation around the z axis.
The Roll is one kind of face distortion that can be easily corrected. A trivial way to correct Roll is to measure the identifies landmarks of eyes, take the centers of eyes, take the line that join the centers of eyes, compute the angle and rotate the face image in the opposite direction.
Yaw is less trivial to address, because in this case we lose part of the face. So, with a moderated Yaw, it is still possible to correct and to create a kind of frontal face; otherwise it's difficult.
Pitch to is difficult to address, because there is a kind of symmetry between the distance from the center of the eyebrows and the nose tip and chin, but this is not necessarily so, because in some people these distances are not that equivalent. So even Pitch cannot be easily corrected.

One way to somehow measure or estimate the quality of a possible response, could be analyze in advance the pose of face in order, for example, and, if it's possible, to discard some frames in the video where face has to high Yaw or Pitch angles.
To measure the quality of a face image, we can for example divide an overall measure of distortions related to both illumination and pose of the face.
SP measure of distortion with respect to frontal pose, expressed in terms of misalignment of roll (α), yaw (β) and pitch (γ):

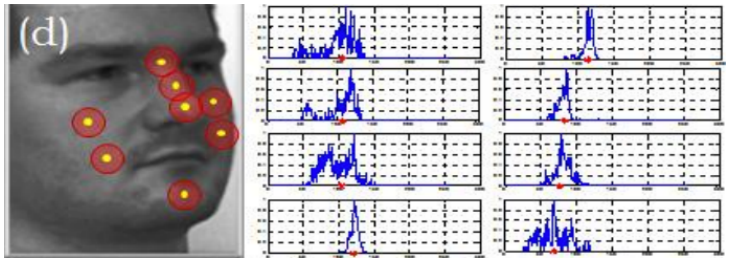
SP = α * (1 - roll) + β * (1 - yaw) + γ * (1 - pitch)

It measure a score related to pose (SP = Score Pose). It takes into account all three possible distortions.



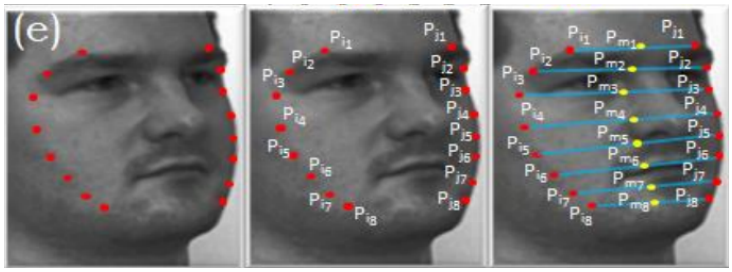
How can we evaluate Roll? We rely on identification of face landmarks. To measure the Roll, we rely on the centers of eyes; we draw the line that connecting the centers of eyes and we measure that angle. Higher is that angle, higher is the amount of Roll.
How can we evaluate Yaw? We rely on the supposed symmetry of face, so that we take again the centers of the eyes and we measure d_l (distance between the center of the left eye and the nose tip) and d_r (distance between the center of the right eye and the nose tip). The more the face is frontal, the lower is the difference between these two elements.
How can we evaluate Pitch? Pitch is again measured considering some relevant landmarks: the distance between the center of the eyes and the nose tip is more or less the same as the distance between chin and the nose tip or the immediate region below it. Again, measuring the ration between these two measures we can suppose how much Pitch is present in the image.
SI is defined as a measure of homogeneity of grey levels in some predetermined face regions:

SI = 1 - F(std(mc))

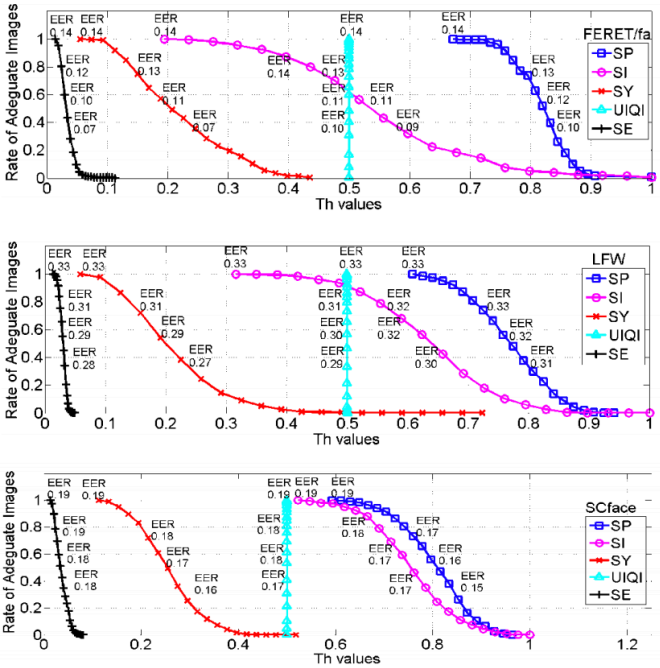


SY is defined as a measure of face symmetry:

$$SY = \sum_{(i,j)\in X} sym(P_i, P_j).$$



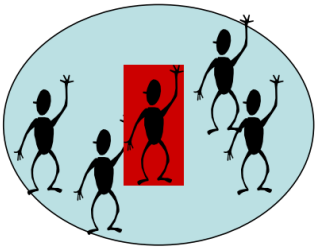
So we can try to measure in advance number of parameters that can help us to discard when it's possible images that are not suitable for recognition and we try to sure in advance a higher accuracy of the system by discarding in advance that images that present lower quality.
On the other hand, we can also apply a kind of wait to the final decision in order to have images with bad quality and the decision based on such images can be accepted only when the score is higher than a threshold that is different from that one that is usually adopted by the system.
We can also try to **measure the "quality" of a quality measure** in order to understand which are the best measures or the better measure to anticipate the reliability of the obtained outcomes. So we may do some comparisons and we may identify how these measures can affect the threshold that is better adopted in order to have a good recognition.
The first test for a quality measure is to check how the values of a dataset are distributed with respect to the returned values.
This allows to understand which is the average level of quality of a face dataset with respect to a specific measure.
Two or more measures can be compared by computing the amount of correlation of the returned values with respect to the images of given dataset.
A measure of quality of input samples allows to discard a-priori (before recognition) those affected by too high distortion which would lead to a wrong response or not reliable from the system.
A further test for a quality measure is to evaluate how it affects the system performance (EER) by varying a tolerance threshold.
A good quality measure must provide error good decrease by discarding as few samples as possible.
We've to take into account two issues: it's not always possible to select Probe (because there are cases where we have only one probe. related); the other one is that, in any case, we must consider the throughput of the system (if we discard to many probes, we may have a much lower throughput of the system).
The EER decreases for each and any score related to quality of pose, illumination and whatever. We can see that how we increase the score that we required from the pose evaluation in order to submit the probe to the recognition system(so the higher is the quality score), the lower is the EER.



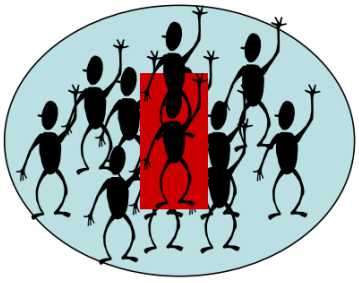
Another approach can be based on the **notion of margin**. The **margin** is defined as the difference between FAR and FRR given a certain threshold.
More related to evaluation than to reliability, but this is an **alternative way to measure the performance** of the system: for each threshold we can take the absolute difference between the two errors.

SYSTEM RESPONSE RELIABILITY (SSR)

In order to measure the possible reliability of an identification response, we notice the major difference between a quality measure for an input sample and a reliability measure for the response of a biometric system. We may also image a kind of chain of three different steps: the first one is to evaluate the possible quality of the probe in order to be able, when it's possible, to just select the best probe (this is something happens before the actual recognition operation); then we have the recognition operation carried out with any kind of algorithm; after this we can evaluate the reliability of the response.
We may have a low reliability for a response even though it was computed over a good quality sample. Because it is a concept that is different from the concept of probe quality: the we can measure the probe quality in advance before the recognition operation; we measure the reliability of the response after the recognition operation in order to evaluate whether we can trust the final decision of the system.
System Response Reliability (srr $\in$ [0,1]) index measures the ability of an identification system to separate genuine subjects from impostors on a single probe basis.
In practice, what we rely on is a notion of "**confusion**" among the possible candidates.
The SRR relies on different versions of function ϕ . We defined and tested two different ϕ functions: **Relative distance** and **Density ratio**.
Both functions measure the amount of “confusion” among possible candidates.
These two measures are not the final system response reliability because, after works, we've to normalize these measures to the respect to different responses of the system.
We have a core function, that is either the Relative distance or the Density ratio, and then the result of this core function is somehow farther processed according to context of the response.
We assume that the result of an identification operation is not only the returned identity but also the whole gallery ordered by distance from the probe, or a short list at least.
We have the list of ordered values. We start from the assumption that we can image a cloud around the returned subject; a cloud means a number of subjects that have distance from the probe quite similar to that of the returned identity.



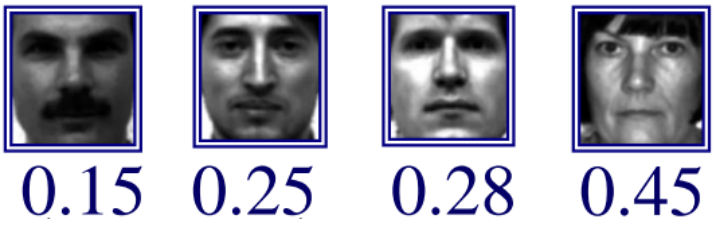
The "**less crowded**" cloud means that there is the lower number of subjects close to the returned identity. That means the cloud of the subjects that have from the probe a similar distance than the returned one.
Less crowded is the cloud, more reliable is the response.
So, consequently, a "**more crowded**" cloud means that the response is less reliable.



Closer are the candidates of returned list, the higher is the possibility of confusion, because it means that they are similar to each other.
Given a probe p and a system A with gallery G, the **Relative Distance** is defined as:

$$\varphi(p) = \frac{F\left(d\left(p, g_{i_2}\right)\right) - F\left(d\left(p, g_{i_1}\right)\right)}{F\left(d\left(p, g_{i_{|G|}}\right)\right)}$$

For this example, we measure the difference between the first returned distance and the second one to see how close are they each other. Then we take the ratio with the respect to the highest distance that we have on the returned list.
In practice, we have in this case the distance between the firsts two is 0.10 and we've on the denominator 0.45.



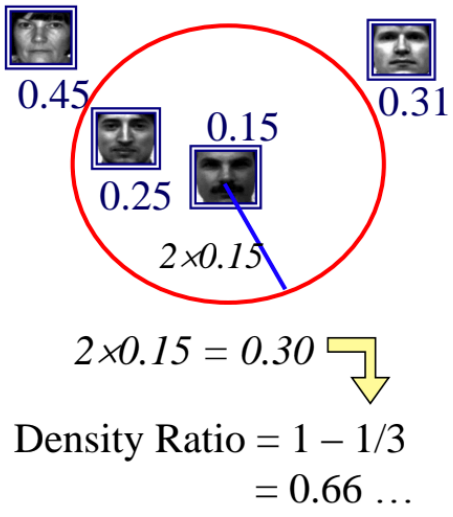
This difference on the numerator is an indirect indication of how g_1 and g_2 are close each other: the closest they are to each other, the lower is the difference.
While on the denominator we've the highest distance from the probe that has been computed either over the whole gallery or the short list the has been returned. So consider that we're measuring the discriminating power of the function with the respect to the submitted probe in relation to the first two returned elements.
If we have a short distance as numerator and a large distance as denominator, we would have a low value for ϕ.
But this low value is not a good sign because it means that with the respect to the higher measured distance these two first elements are much closer to each others with the respect to the farthest distance from the probe.
So **lower is this ϕ, worst is the reliability of the system**.
We now consider that there is a critical value for ϕ_k that depends both on the gallery and on the recognition algorithm, so it is must be somehow computed in a kind of pre-operation face, because for any gallery and for any recognition algorithm we can identify a critical value of ϕ (ϕ_k) that has a similar meaning of EER. Even if they are not exactly the same, because we don't want to have the same probability of errors but we want to minimize the wrong estimate of ϕ(p): encroach recognition that are erroneously taken as reliable due to the value of ϕ and correct rejections that are erroneously refused due to the value of ϕ. **Higher will be ϕ(p), the better**, because this means that the difference between the first distance and the second distance is very close to the maximum absolute distance.
Then we can identify this ϕ_k as the critical value that somehow divides the incorrect answers that were taken as reliable due to the value of ϕ, from the correct answers that were incorrectly considered as not reliable due to the value of ϕ.
Given a probe p and a system A with gallery G, the **Density Ratio** is defined as:

$\phi(p) = 1 - |N_b| / |N|$

With:

$$N_b = \{g_{i_k} \in G \mid F(d(p, g_{i_k})) < 2 \cdot F((d(p, g_1)))\}$$

We count how many feature vectors on the list have a distance from the probe that is less than twice the first distance. So we go to count on the short list returned what is the number of templates with the distance that is less than twice the first one.



This function is **less sensible to outliers**, and in fact **usually performs better than Relative Distance**.
We're going to farther on the candidate list to see what happens even after the second one possibly. In the sense that we see how much far is the first identity from the other returned ones. Higher is the number of identities computed over the all returned list with the respect to twice the first distance, more crowded will be this cloud and so lower will be the reliability of the system.
So higher will be ϕ(p), better will be the reliability of the system.
As a **drawback**, its definition takes to consider narrower clouds when the first retrieved identity is closer to the probe. On the contrary, a large distance takes to a larger cloud, which can be expected to be more crowded in any case. Any attempt to substitute 2 with an adaptable parameter did not achieve better results.
We need to identify a value fostering a correct separation between wrong rejections of enrolled subjects and wrong recognitions of not enrolled ones, both supported by the reliability value.
The critical ϕ_k is given by that value able to minimize the wrong estimates of function ϕ(p), i.e. not enrolled subjects erroneously recognized with ϕ(p) higher than ϕ_k, or genuine subjects wrongly rejected because recognized with ϕ(p) lower than ϕ_k.
The distance between ϕ(p) and ϕ_k is significant for reliability.
Farther is the obtained ϕ from ϕ_k, the better because it means that we're far from the critical point. This distance has also to be considered with the respect of the maximum distance that we can have between the obtained ϕ and the critical point ϕ_k according to the axis.
We also define as the width of the subinterval from to the proper extreme of the overall [0,1) interval of possible values, depending on the comparison between the current ϕ(p) and:

$$S\left(\varphi(p),\bar{\varphi}\right)=\begin{cases}1-\bar{\varphi} & if \quad \varphi(p)>\bar{\varphi}\\ \bar{\varphi} & otherwise\end{cases}$$

If ϕ(p) is higher than the critical value ϕk then we take as a possible range over which ϕ can get values with the respect to the critical value as 1 - ϕ_k; otherwise we're going to below and this means that we take as normalization factor exactly ϕ_k.
At the end, the **final value for the system response for reliability (SRR)** is the difference between the obtained value of ϕ(p) and ϕ_k and this function that provides the maximum that could be achieved.

$$SRR=(\varphi(p)-\bar{\varphi})/S(\bar{\varphi})$$

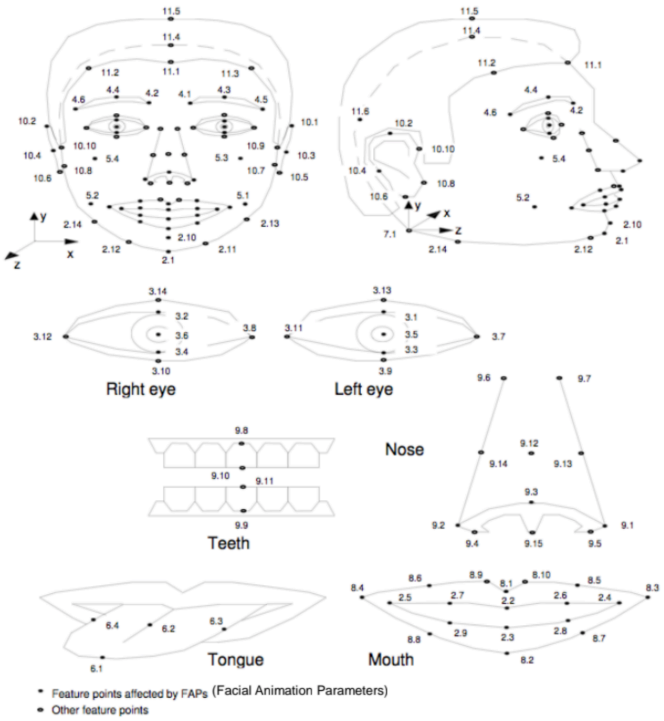
The **reliability threshold** th can be automatically estimated by exploiting a certain number M of successive observations.
After computing a reasonable formulation for the system response reliability we can also set up a threshold in order to evaluate when it is actually convenient to reject even an apparently correct identification due to the fact that we don't consider it reliable enough.
So now we've two thresholds: one that determines false acceptances with the respect to verification and identification and the other one for the reliability.
Once we've a value that meets the acceptance threshold can we also consider the system response reliability?
Yes, because this means that we can also refuse an identification if we don't trust it.

We would desire to have a high average (the system is generally reliable) and a low variance (the system is stable).
We can summarize as:

$$th_i = \left| \frac{\mathrm{E}[\overline{S}_i]^2 - \sigma[\overline{S}_i]}{\mathrm{E}[\overline{S}_i]} \right|$$

FACE RECOGNITION: INTRODUCTION AND FACE LOCALIZATION

The most important factors influencing the feasibility of a biometrics are Accuracy/ Reliability and Acceptability.
DNA recognition is the most accurate trait, but also one of those requiring the most intrusive procedures. Fingerprints are accurate and more easily accepted, but require an aware and collaborating user. Moreover in some cases they might be of low quality (e.g., hard workers).
Face has a high acceptability, the user may be unaware of image capture, but accuracy is still to be improved.
It is natural to recognize a person from face and it is usual for people to be photographed (e.g., fingerprints may be associated with the bad feeling to be suspected to be a criminal).
It has an extremely high recognition rate in controlled conditions.
Acquisition devices are easy to deploy in the operation environment.
It is possible to integrate it in logical access (login).
It is possible to integrate it in remote control applications.
However it's to consider that **face is a complex object**. For example there are the MPEG-4 feature points that are used to synthesize the face and also to reproduce the most natural possible movements of the difference landmarks. The Facial Animation Parameters FAP handled by MPEG-4 can express the complexity of face, because each and any such point been used in synthesizing real faces is also the representation of a point in the real face that can change its relative configuration during an expression change.
Let's compare face with others biometric traits.The classical properties that can be taking into account are: Universality (everybody must posses the trait), Uniqueness (if we want a strict recognition), Permanence (must not change in a medium time at least), Collectability, Performance (the accuracy that we can achieve with the available



MPEG-4 feature points

methods), Acceptability and Resistance to Circumvention (how difficult to spoof the identity of another person by using that biometric trait).
Face, with the respects to the other traits has an High Values for Universality, Collectability (it's sufficient to take a photo) and Acceptability. Face has a Medium Value for Permanence, because in a certain edge range the face doesn't change. And finally Face has Low Values for Uniqueness (this refers to the possibility to have similar traits for relatives for example), Performance and Circumvention.

Biometrics	Univer-sality	Unique-ness	Perma-nence	Collect-ability	Perfor-mance	Accept-ability	Circum-vention
Face	H	L	M	H	L	H	L
Fingerprint	M	H	H	M	H	M	H
Hand Geometry	M	M	M	H	M	M	M
Keystroke Dynamics	L	L	L	M	L	M	M
Hand vein	M	M	M	M	M	M	H
Iris	H	H	H	M	H	L	H
Retina	H	H	M	L	H	L	H
Signature	L	L	L	H	L	H	L
Voice	M	L	L	M	L	H	L
Facial Thermogram	H	H	L	H	M	H	H
DNA	H	H	H	L	H	L	L

H=High, M=Medium, L=Low

Biometric Comparison Chart

Sr. No	Characteristics	Fingerprint	Iris	Face	Palm Print	Voice Recognition
1	Speed	Medium/Low	Medium	Medium	Medium	Medium
2	FTE Rate	Medium/Low	Low	Low	Low/Medium	Medium
3	Standards	High	Medium	High	High	Low
4	Uniqueness	High	High	Medium	High	Medium
5	Maturity	High	Medium	Medium	Medium	Low
6	Durability	High	High	Medium	High	Low
7	Invasiveness	High	Medium	Low	High	Low
8	Overtness	High	Medium	High	Low	Low
9	Range	Low	Low	Medium	Low	High
10	Template Size	Medium (250-1,000 bytes) (per finger)	Medium (688 bytes)	High (84-2,000 bytes)	Medium (250-1000 bytes)	High (1,500-3,000 bytes)
11	Age Range	High	High	High	High	Medium
12	Universality	High	Medium	High	High	High
13	Stability	High	High	Medium	High	Low
14	Skill	Medium	Medium	Low	High	Low
15	Accuracy	Medium-High	High	Medium	High	Low
16	Hygienic Level	Low	High	High	Low	High
17	Performance	High	High	Low	High	Low
18	Cost	Low	High	Low	High	Low

FTE means Failed To Enroll.
Face Recognition can be used in Forensics, in logic access (the person is not physically enter in a place), border control and also in a crowd given special situations.

PROBLEMS

The main problems that we may have in Face Recognition are the **Intra-personal Variations** (pose, illumination and expression are all factors that can increase the amount of variations even among templates of the same person) and the **Inter-personal Variations** (it's not necessary to have relatives to have similarity that is quite expected in that case but it is also to find out very similar persons that have nothing to do each other). Also the effects caused by PIE (Pose, Illumination, Expression) variations, we have to take into account the effect caused by the aging, especially with some kind of applications that have much to do with forensics. But it's so difficult to find the so called longitudinal dataset; this is a dataset where it's possible to collect face images of a person from childhood to adult age (if the person is collaborative).
These are some **popular databases** that can be used for experiments:

- AR-Faces
- FERET
- MIT
- ORL
- Harward
- MIT/CMU
- CMU test set II

Datasets are evolved in time because whenever a problem is somehow solved in some satisfactory way, then new problems arises and researchers try to solve these new challenges.
All those databases are collected so that each image contains single face in a more or less central position at a good distance (close to the camera).
The size of images is important to know because larger the size better the resolution. However, in many case it's even desirable to reduce the resolution of an image because the interesting details can still be detectable and computational demand decreases together with the image size.
In these all datasets there are some kinds of variations related to the PIE factors. However such variations are somehow controlled in the sense that in order to better evaluate the ability of the system in different conditions images are captured in variables but controlled conditions so that the researchers know that in ground truth image for example the person was in profile or that he had a yaw angle of a certain degree.
So in this sense, this is not a pure uncontrolled conditions because we know exactly which is the distortion.

FERET

FERET was acquitted in different versions. The database contains:

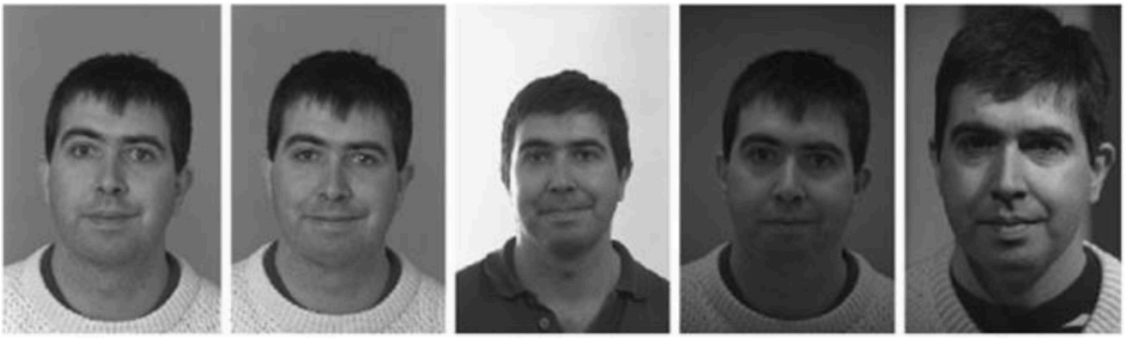
- **Pose variations**: the subject is captured under different orientations, each denoted by a pair of letters in the image name. There is a camera and the subject is on a chair can rotate in different positions with respect to the axis of the camera.
- **Illumination variations**: variations are not available for all subjects and are not underlined in image names.
- **Time variations**: subjects are captured at different times in different sessions.

The database contains a total of 14051 images divided in different categories.
The authors provide a set of lists related to the names of the images for each category.
A set of files contains the positions of eyes and mouth for each image.

Why is it important to have the positions of the eyes? Because the positions of the eyes can be used in order to normalize face images, so that we can have all images of the same size.
If we process images of different size, this influences the relative positions and so the geometry that we can extract.
So it's important to be able to normalize all images to a similar size and this can be obtained by using as normalization factor the interocular distance.

Two letter code	*Pose Angle (degrees)*	*Description*	*Number in Database*	*Number of Subjects*
Fa	0 = frontal	Regular facial expression	1762	1010
Fb	0	Alternative facial expression	1518	1009
ba	0	Frontal "b" series	200	200
bj	0	Alternative expression to ba	200	200
bk	0	Different illumination to ba	200	200
bb	+60	Subject faces to his left which is the photographer's right	200	200
bc	+40		200	200
bd	+25		200	200
be	+15		200	200
bf	-15	Subject faces to his right which is the photographer's left	200	200
bg	-25		200	200
Bh	-40		200	200
bi	-60		200	200
ql	-22.5	Quarter left and right	763	508
qr	+22.5		763	508
hl	-67.5	Half left and right	1246	904
hr	+67.5		1298	939
pl	-90	Profile left and right	1318	974
pr	+90		1342	980
Ra	+45	Random images. See note below. Positive angles indicate subject faces to photographer's right	322	264
Rb	+10		322	264
Rc	-10		613	429
Rd	-45		292	238
Re	-80		292	238

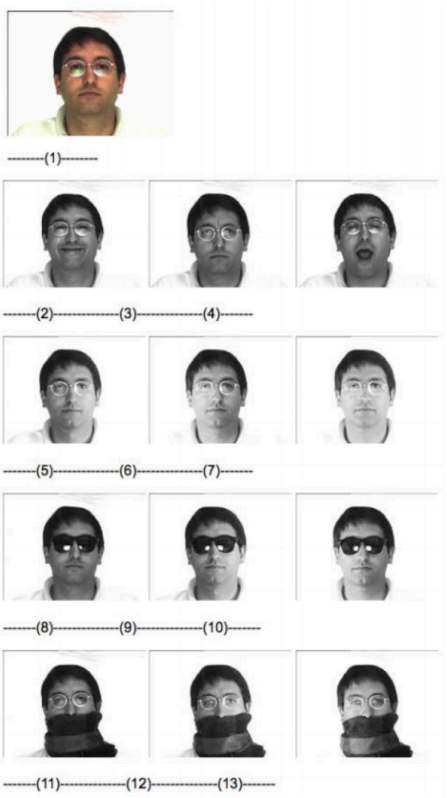
This is an example of how a database should be collected: with images of the same subject with different conditions.



AR-Faces

In **AR-Faces** we've different expressions like **Neutral Expressions**, that is usually what we put in the gallery (because in general we can assume that the enrollment happens in controlled conditions and so we have good quality images used for enrollment).
Then we also have **Smile**, **Anger**, **Scream** as expression variations.
We have also **different light sources** that illuminate the face in different points of view.
There are also images where people wearing sunglasses with a certain kind of light and wearing scarf with a certain light.

This database is quite interesting but it has a huge limitation that is the homogeneity of the background. During face detection, one of the problems that we may have is having a noisy background. So background, or other elements in the image, can hinder or make difficult the face detection.
There are 126 people (over 4,000 color images). Two sessions per person (2 different days).



CMU-PIE

We have a huge equipment that allows to simulate both pitch, by increasing the height of the camera with the respect to the person, and yaw, by changing the angle of the camera.
There are 68 subjects and 608 color pictures per subject.
Each person is imaged under 13 different poses, 43 different illumination conditions, and with 4 different expressions.

STRUCTURE OF A FACE RECOGNIZER

The **first step** is the face image capture. The capture, according to the distance, may only contains the face or a large environment.This is the case why the image is first enhancement.
The **image enhancement** refers for example to increase the sharpness of an image, to try to eliminate some blurring when possible, to increase the contrast and so on.
After this, we've the detection and localization. We often refer to detection and localization as if they were more or less the same operation, but there is a difference.
In general **Detection** only returns a binary response: yes, this object is in the image; no, this object is not in the image. **Localization** also provides the exact position of the element within the image.
Once we have localized or detected the face, we may also crop the **region of interest** (**ROIs**) in order to decrease the processing time by limiting the following processing step only to the relevant regions.
It's also possible to hierarchically continue this process in the sense that many approaches stop once they've found the region of interest containing the face; but we can also go farther by identifying different patches. Consider that patch based approaches in face recognition have become very popular because they consider a partition of face images so that there are better able to take variations due the occlusions or to an even illumination.
Finally, we have feature extraction and template construction (**biometric key**).
When we have **Face Localization**, the problem that we want to tackle is: given a single image or a video sequence, we want not only to detect the presence of one or more faces but also locate their exact position within the single image.
Why the exact position is important? Because the exact position is used to identify the region to crop, so that the following computation can take only into account that region of interest.
It's necessary that the Face Localization that we adopt is independent with respect to position (we cannot always assume that the face right in the center of the image), orientation (pose), scale (we must be able also to detect small faces in a big image), expression, possibility that this element can be different for different subjects in the image, as well as to illumination or cluttered background.
There are others problems that can be raised by the fact the we have many subjects, the fact that we have subjects at different scale and we may also have a cluttered background, in the sense that we may have false detection when we have some particular configurations that may suggest the presence of a face but it's not present there.
Another problem is: **when can we be sure to have detected a face?** Because our algorithm can also provide false detections that we must try to correct as much as possible.
Can Face Localization be fooled? Yes. With the makeup or with other elements on our face, it's possible to hide from the Face Localization. This is called “**Adversarial Fashion**”.

SOME APPROACHES

Some approaches for Face Localization can be based on **special properties of pixels belonging to the face**. For example edges, in particular positions, or skin color.
We may rely on special face geometry properties, like Constellation or Landmarks and so we look for specific geometry features that relate this Constellation (Feature searching).
Or the template matching against a standard model: Correlation, Snakes and Active Shape Models.
We can also have **Image-based Techniques** that address the localization problem as a generic pattern recognition problem (class of faces). So we have the class of faces; we want to see if our image has elements that fall into this class.
Then we have **Support Vector Machines**, **Neural Network** or **Hidden Markov Model**.
Recent advances use exemplar or ready stored for learning with special distributions of Constellation of landmarks in order to use the exemplars and to see if an incoming images matches some of these reference exemplars.
These approaches are based on Machine Learning strategies that rather try to build a model of the object by using a training face that is usually the most computationally demanding step. The more images we've as examples to submit to the system, the better is the training.
A practical example of face localization is the method that used the algorithm devised by Hsu, Mottaleb and Jain.
The method proposed by them is composed by two macro phases: **Face Candidates Detection** and **Face Candidates Verification** through facial features detection.
In general, it's often the case that in order to detect an object, we first look for possible candidates regions in the image that may contain the object that we're looking for. So it's a general pattern to have a Candidates Detection first (in this case a Face Candidates Detection).
Then a Face Candidates Verification is carried out through the detection of expected facial features.
So we first identify candidates regions using more general characteristics and then we go in more details by checking final details in order to finalize the detection or discard the region.
Even regarding the face region candidates identification we've a **number of sub-steps**.
So in candidates region identification we've to first **pre-processing the face**: this means that we've a series of steps and procedures that are applied to the image in order to enhance some image characteristics that can be useful for our task.
In this face detection approach the first pre-processing step is the **Illumination Compensation**. According to different sources of light and to different illumination conditions in the same environment, the visual rendering of color may change.
Then, there is the **Color Space Transformation**. This is an important step in general in image processing especially when we want to identify regions with the same color characteristics. This is also called **Segmentation or Color Clustering** in image processing.
This is useful when we want separate the regions that potentially contains the skin from a darker region that could be the mouth or the hair line. However, the most used color space, the **RGB**, **is not suited** for many of such tasks. This because, the same visual rendering can corresponds to different combinations of red, green and blue. So, since the computer doesn't consider the visual rendering, because it has not any interpretations of colors but it interprets the numbers that make the colors that we see, that colors that appear similar to our perceptual system, may be expressed on the digital image by different combinations of RGB channels.

So this means that we may have the so called Over Segmentation Process: groups of pixels that actually from a perceptual point of view should belong to the same region, can be splitted in different regions just because for some sensors characteristics they have been expressed by different combinations. The same can happen on the contrary.
So colors that are quite close on the RGB channels may be considered similar even due they appear different. This happens because RGB is not a perceptual space and so it doesn't follow the perceptual rules that allows us to distinguish colors, but it is a kind of mathematical model based on the combination of basic colors.
Then, we've the last pre-processing step, that is **Localization Based on Skin Model**.
Skin Model is something very crucial in this case, because we must device a skin model that is able to encompass all possible nuances of human skin. For example, the ethnic origin of a subject may influence the appearance of skin.
After that, we've the **Localization of the Main Face Features**, like eyes, mouth and face contour provided by landmarks.

- **Illumination Compensation**
  The skin-tone depends on the scene overall illumination. It's not depend only on this, but also on sensors that have captured the face image.
  The illumination compensation uses «**reference white**» to normalize color appearance. The way to find the Reference White is standardized and this Reference White is used to shift the overall color scale in order to take this reference point into account.
  What is the Reference White? First of all we look for the **luma value** for each pixel. Luma is the weighted sum of **gamma-compressed RGB components** (denoted as R'G'B'). So we start from the single values over the RGB channels and we transform each value with a gamma-parameter that is lower than 1; in this way we obtain the so called gamma-compressed RGB colors.
  After we have transformed the image according luma values, we take the pixels with the top 5 percent of luma. In practice, we take the luma for a decreasing frequency and we take the pixels with the top 5% of luma values.
  They must be a sufficient number with the respect to the image size, because they must represent a good reference point for the automatic scaling of all the other colors.
  If a sufficient number of reference white is found, or the average color is not similar to skin tone, the RGB components are adjusted so that the average grey level of the reference white is linearly scaled to 255 (that is the with in RGB). So the Reference White becomes that value and everything is scaled linearly according it.
- **Color Space Transformation**
  RGB is not a perceptually uniform space, so colors which are close to each other in RGB space may not be perceived as similar. Colors similar to each other may appear with different numbers. Since RGB can be expressed as a vector of three components, if we for example rely on the euclidean distance, we may have colors that apparently have high distance in the euclidean space of the three channels but actually they are perceived as similar. In a clustering operation this can be destructive: there is an over segmentation because pixels belonging to a certain region are rather classified as they belonging to an other region.
  The Skin model is a set of close colors (cluster) within the color space. It is advisable to perform the detection in a different space.
- **Localization Based on Skin Model**
  It entails **Variance-based Segmentation** and **identification of Connected Components**.
  In practice, there are some general rules that are more or less applied to all segmentation strategies, but in general many segmentation approaches rely on the knowledge of some characteristics of the object that we're going to process.

The simplest method of image segmentation is called the **thresholding method**. "Thresholding" means everything and nothing, because there are a lot of ways to apply thresholding: fixed thresholding, adaptive thresholding. In any cases the bases of all this method is to identify one or more thresholds (threshold is a value belonging to the range of colors in this case).
The key of this method is to select the threshold value (or values when multiple-levels are selected).
How can we use threshold for segmentation? For example, if we've a very gray image and we want to transform it into a black and white image because we know that some regions may belong to a certain kind of object. In practice, we want to create a mask from this black and white image. What is the most trivial way to create such mask? To choose a threshold between 0 and 255 and see what happens if we transform into black whatever is below the threshold and into white whatever is above the threshold.
When we have a certain kind of images, it's difficult to exploit different threshold for each image. Because this would entail to have a pre-analysis of distribution of colors that would increase the processing time. So what happens is that there is a kind of training face where we experimentally observe what happen for a large number of images, by fixing threshold in a certain way.
Several popular methods are used in industry including the maximum entropy method, Otsu's method (maximum variance), and k-means clustering (the problem in this method is that that k must be known in advance).
Conventional Variance-based Thresholding method: we start from the min values in a certain region and then we compute also the variance that is the difference of each pixel from the average.
We have a gray scale image with L gray levels G = [0, 1, ..., L-1]; M x N is the size of the image; an image can be considered as a kind of function f(x, y) that fills an array of M x N dimension and each cell (pixel) of this array contains the gray value of the pixel at point (x, y). The number of pixels with gray level i is denoted by n_i.
We can represent this as an **histogram**: in the sense that the histogram has on the x axis the 255 bins and on the y axis we have a bar denoting n_i. This is how we obtain this kind of distribution (**frequency**).
In order to have a **comparable histogram**, what is usually done is to divide this to the size of the image, because this provide us a comparable value between 0 and 1, so the frequencies become probabilities. Moreover, this makes the histogram of images with difference size comparable.
Once we've this distribution of probability of the gray levels within the image, we can start from the assumption that the pixels in the image can be divided in **two classes C_0 and C_1** and that such two classes should be divided by a gray level t: pixels with a gray level below t are collected into C_0 class and pixels with a gray level above t are collected into C_1 class. So, at the end, C_0 should be the set of pixels with levels from 0 to t and C_1 should be the set of pixels with levels from t+1 to L-1.

$$p_i = \frac{n_i}{M \times N}, \quad p_i \geq 0, \quad \sum_{i=0}^{L-1} p_i = 1$$

We in general assume that C_0 and C_1 should correspond to the object class and background (or vice versa) and so let's compute separately the probabilities of each class. The aggregate probability of a class will be given by summing up the values of pixels that have a value below t (for C_0 class); summing up the values of pixels that have a value above t (for C_1 class).

$$\omega_0 = \sum_{i=0}^{t} p_i, \quad \omega_1 = \sum_{i=t+1}^{L-1} p_i$$

In any probability distribution, we can also find the mean probability or the variance of probability.
In this case, we can find out the **mean gray level** for each of the two classes: regarding class C_0, we will sum up the probability of pixel with value i by i, dividing by the probability of the whole class; the same happens for the class C_1, of course taking the pixels with values above t.
We can also compute the **class variances**: according to definition of variance, we take for each pixel the value for the pixel minus the average of the class; after that we square the value and multiply it by the probability of value i; then we divide by the probability of class. The same happens for C_1 class.
We can compute in Otsu method the within class variance as the weighted sum of the class variances wighted by the probability of each class.
How can we use this in order to find out the optimal threshold?
We want minimize variance in class 0 and minimize variance in class 1. So we want look for a threshold that is able to minimize the variance within each of these two classes. This may happen by changing the threshold t, because if we change it we may have a better distribution of values around the new average for a certain class.

$$t^* = \arg\min_{t \in G}[\sigma_w^2(t)]$$

In general, what we do is to process a number of exemplars of the problem that we want to tackle and to find outs once and forever a threshold that can provide a good performance in most cases.
Let's say that we use a kind of multiple level segmentation in order to detect the skin tone pixels; so we use local color variance and then we try to find out the so called **Connected Components**.
A Connected Component is a region with no hols or a region such that from each pixel is possible to reach any other pixel without exiting from the region.
These connected components are grouped according to spatial closeness and similar color and these represent the face candidates.
Once we have this homogeneous region thanks to **morphological operators** that allow us to reach this result (operators like erosion or dilation that help to close the holes), we can start with the eye localization.

- **Eye Localization**
  The algorithm builds two different eye maps: one based on **chroma** and the other one on **luma**.
  The creation of **Chrominance map** relies on the observation that the region around eyes is characterized by high values of component blue C_b and low values of component red C_r. This because there is a kind of concavity around the eye, so that it's easy to image that the blue (dark) component is higher than the red component.
  So the **Eye Map** is computed by taking for each pixel the C_r and C_b values.
  Then we use the following formula where C_r tilde is the complement with the respect to the pure C_r:

$$EyeMapC = \frac{1}{3}\left\{\left(C_b^2\right)+\left(\tilde{C}_r^2\right)+\left(\frac{C_b}{C_r}\right)\right\} \qquad \tilde{C}_r = 255 - C_r$$

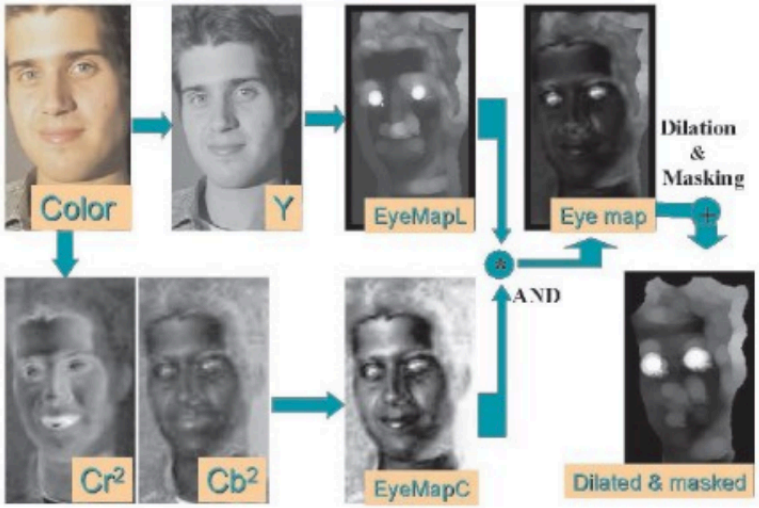
where C_b^2, $(\tilde{C}_r)^2$, C_b/C_r all are normalized to the range $[0, 255]$

So for each pixel in that identified region we compute this value and also compute the Luminance Map.

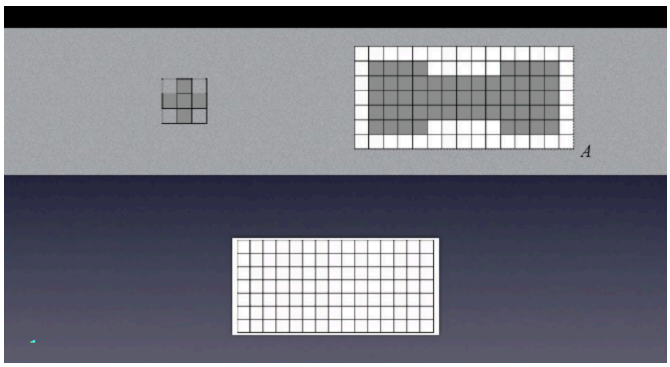
The basic consideration in the creation of **Luminance Map** is that the eyes usually contain both light and dark zones that can be highlighted by morphological operators (dilation and erosion with hemispheric structuring elements).

$$EyeMapL = \frac{Y(x,y) \oplus g_\sigma(x,y)}{Y(x,y) \ominus g_\sigma(x,y) + 1}$$

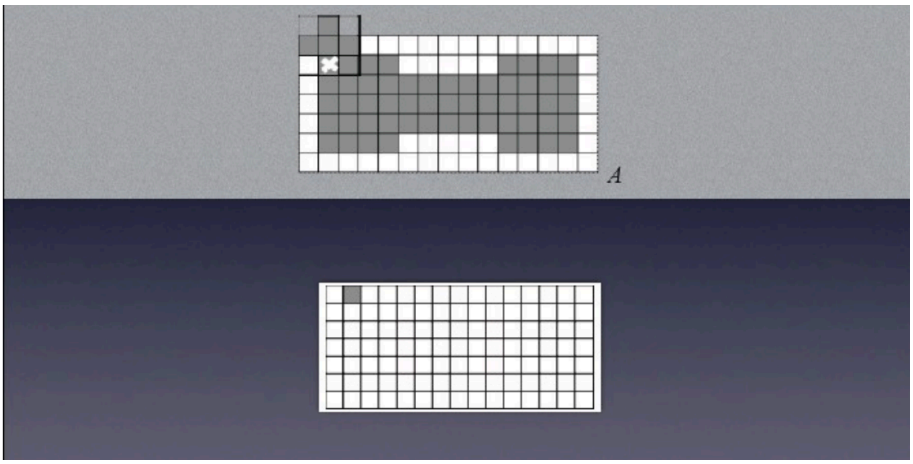
The operator in the numerator is the dilator, while the operator in the denominator is the erosion.
We can observe that the Chroma map is enhanced by histogram equalization (that is another basic operation in image processing and means to try to somehow lower the contrast among the different regions in order to maintain a kind of equiprobability of all pixels).
Once a Chroma map has been processed through histogram equalization, the two maps are combined through AND operator.
The resulting map undergoes dilation, masking and normalization to discard the other face regions and brighten eyes.
Further operations allow to refine this map.

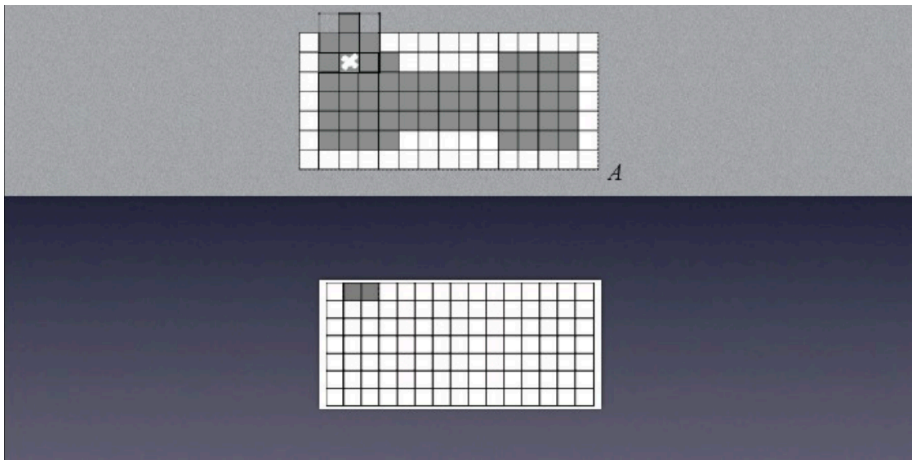


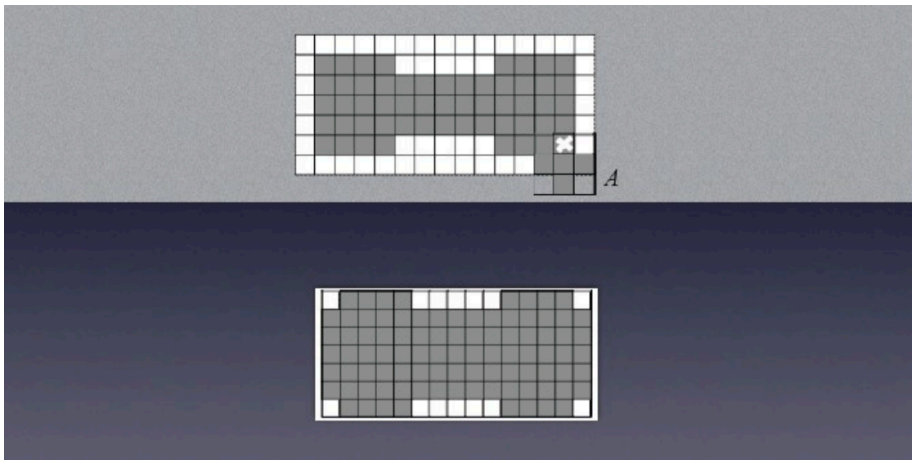
How does Dilation work?
The **dilation operator** takes two pieces of data as inputs. The first is the image which is to be dilated. The second is a (usually small) set of coordinate points known as a structuring element (also known as a kernel). It is this structuring element that determines the precise effect of the dilation on the input image.
The **mathematical definition of dilation for binary images** is as follows: suppose that X is the set of Euclidean coordinates corresponding to the input binary image, and that K is the set of coordinates for the structuring element; let K_x denote the translation of K so that its origin is at x; then the dilation of X by K is simply the set of all points x such that the intersection of K_x with X is non-empty.
So what is the dilation of X by K? Is the set of all points x where in turn K has been centered, such that the intersection of K_x with X is not-empty.



In the previous figure, A is the image, X is the set of points of the image and the structuring element is a 3x3 window with a cross shaped pattern of pixels.
We slide this over the image A and we take, according to the definition, the set of points such that the intersection of K_x with X is not empty. x pixels that once K is centered on these pixels provided a non-empty overlap with the kernel, if they were white they become black and if the they were black they become white. If we center K over x, if there are any intersection with black pixels in the image and the pattern in the kernel, then that x will become black (whatever was its original color), otherwise it will become white (whatever was its original color).







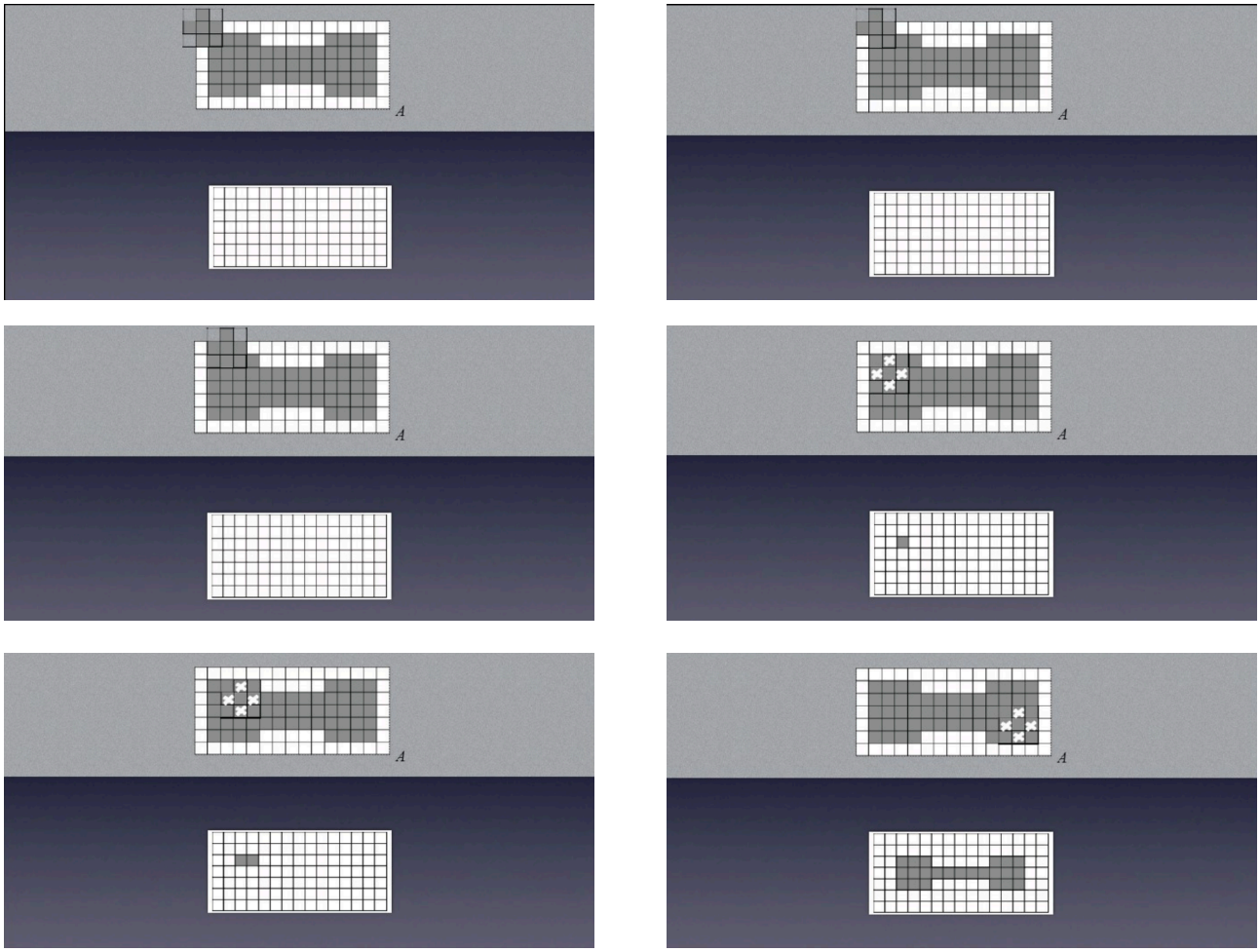
How does Erosion work?

Erosion is the opposite of the Dilation.

The **erosion operator** takes two pieces of data as inputs. The first is the image which is to be eroded. The second is a (usually small) set of coordinate points known as a structuring element (also known as a kernel). It is this structuring element that determines the precise effect of the erosion on the input image.

The **mathematical definition of erosion for binary images** is as follows: suppose that X is the set of Euclidean coordinates corresponding to the input binary image, and that K is the set of coordinates for the structuring element and let K_x denote the translation of K so that its origin is at x; then the erosion of X by K is simply the set of all points x such that K_x is a subset of X.

Being a subset of X means that all points that appear as black in the kernel must be black also in the original image. Even if a single point that is black in the kernel, is white in the original image, then the centering pixel becomes white. In practice this means that all the pixel of the structuring element must be black pixels in the image.

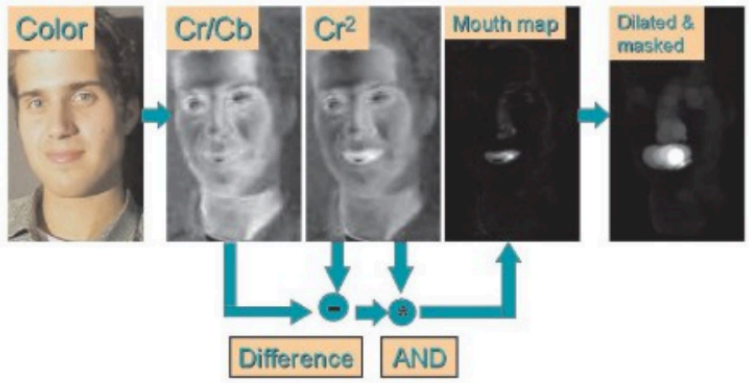


- **Mouth Localization**

  In the mouth region the C_r component is higher than the C_b one; the response to C_r/C_b is low, while the response to C_r^2 is high:

$$MouthMap = C_r^2 \cdot \left(C_r^2 - \eta \cdot C_r / C_b\right)^2$$

$$\eta = 0.95 \cdot \frac{\frac{1}{n}\sum_{(x,y)\in FG} C_r(x,y)^2}{\frac{1}{n}\sum_{(x,y)\in FG} C_r(x,y)/C_b(x,y)}$$



- **Face Contour**
  The algorithm analyzes all the triangles composed by two candidate eyes and a candidate mouth.
  There are a **number of checks** that are applied; so that we consider some kind of expected **variations in Luma and orientation of the gradient** that contains the triangle with eyes and mouth. The gradient is the direction of the color variations among neighboring pixels.
  We also check the **geometry and orientation of the triangle**.
  And finally we check the **presence of a face contour** around the triangle.
  In this way we can assign a score to each candidate that satisfies the conditions and also select the triangle with the highest score.

Algorithm Viola-Jones

It's also based on a Machine Learning strategy so that in this case we can better stress the role of a good training face on the final performance of the system. It's efficient on good quality (from the point of view of pose) images.
The algorithm is **image based** and can be applied to face detection (but also to eye and mouth detection in a hierarchical strategy). In this case we don't explicitly look for eyes and mouth, so we can also consider it as a global approach that uses specific features that can be found. We use general features that are not necessarily limited to face recognition.
For example, once a new training is performed in order to look for other kinds of object, the same algorithm can be used also in a hierarchical way (once we've detected the face) in order to detect the eye region and the mouth region.
The algorithm **requires to create a classifier** that is initially trained using multiple instances of the class to identify (positive examples), and several instances of images that do not contain any object of the class but may cause an error (negative examples).
So, if we're training the algorithm in order to detect faces, we must have a huge amount of faces over which the training can be carried out. If we're looking for mouth, then we will have a huge amount of images just centered on the mouth or cropped over the mouth.
These are called **Positive Examples** because in practice they help the system to learn which are the stereotypical elements that make up a face.
The significant and relevant presence of **Negative Examples** is important as well.
What is a good Negative Example? We may even want to train the algorithm over a different kinds of objects or even refine the behaviour of the existing one since we can even re-train the system starting from the last results published in the available models.
The good Negative Examples must be at least as many as the Positive ones; they are not

samples that contain the object that we're looking for but they could fool the system. So they are samples that could be wrongly classified.
We must stress the training of the system by having the system learn from the images that can fool it.
Training is designed to extract several features from the examples (so the training tries to model which are the most relevant elements to look for in order to detect the face) and to select the most discriminating ones (elements that can better suggest the presence of the face respect to other kinds of geometrical elements). The statistical model which is built incrementally contains such information. Because in this case training is not a one shot process, but it is iterated: there is a kind of learning threshold that somehow we consider feasible in order to assume that the training can be considered as completed.

Which are the relevant elements in this case?

Misses (a present object is not detected) or **false alarms** (an object is detected but it is not present) **can be decreased by retraining** adding new suited examples (positive or negative).
Misses are the false negative images where a present object is not detected. The count of misses can also depend non the kind of images that we have submitted to the system. Because when we have for example a single face per image, each image is either a possible miss or a false alarm.
False alarms is when an object is detected but it is not present.
The count can be a little bit more complicated but can be easily extended to the case when we have images with more instances of the object that we're looking for and so in that case whenever the number of detected object is lower than the expected number (always consider that in the evaluation face we've the ground truth), we can count the number of misses.
On the other hand, a false alarm is whenever a region of interest is identified that is not a face.
In this kind of evaluation is important to have an accurate annotation of the ground truth in terms of true regions of interest containing faces and also is important to accurately consider the result of the system in the sense that is not just sufficient count whether the number of regions of interest is equal to the number of expected faces. This because some of such regions of interest could be false alarms while a true face is missed.
In this case we focus on a **Face Classifier**. Viola-Jones uses **Adaboost approach**.
The detector is a face classifier that is able to associate an input pattern with one of two classes face/non face.
So what we're training in this case is a binary classifier. As in any Machine Learning problem, training is slow, but detection is very fast.

Which are the key ideas?

- **Integral images**: Integral images are used for fast feature evaluation. A **feature evaluation** means: for each sub-region of the image that is taken into account, the system determines whether a certain feature is present in that region or not.
- **Boosting** for feature selection: the **Adaboost approach**. Once we've detected the presence of the feature, we've to decide whether such detection is relevant or not and how it is relevant in order to detect the final object.
- **Multiscale detection**: in the multiscale detection we repeat the overall process over window images of increasing size in order to be able to detect the object both if it appears in small size with respect to the overall image or whether it occupies most of the image as in a closeup photo.

Localization is performed by a **search sliding window** (with varying size) over the image. The window is classified as face/non face according to the features extracted from it.

Each time the window is centered in a region of the image a certain computation is carried out in order to detect the presence of a certain feature.

What is the advantage of the method?

Once we don't find the first features to look for in the window we immediately discard the window, so it's not necessary to arrive at the end of the chain of features. That's why the **order is important**; because we've a kind of cascade decision and in any kind of cascade decision if we miss something in one step, we will never recover it in the following.

AdaBoost learning procedure

This is not limited to be used in the **Viola-Jones** algorithm but it's a general Machine Learning approach.

Boosting is a classification scheme that works by combining **M weak learners** (**linear classifiers**) to obtain a strong one $H_M(x)$ (**nonlinear ensemble classifier**).

In practice a strong classifier is not a new classifier actually but it is simply the correct sequence of weak classifiers each applied with the correct weight (in order to correctly weight the decision).

What is a linear classifier?

If we imaging the feature vectors that characterizing a certain object as being spread in a space where each feature represents a dimension, the points representing each class can be more or less overlap.

A **linear classifier** is a classifier that tries to divide the two or more classes using only linear elements.

Linear classifiers are not always able to clearly separate classes. Especially when we've points in one class that can intrude in the other class.

How can we describe the final strong classifier? It's simply the sequence of weak ones.

What is a weak learner?

A **Weak learner** is a classifier with a possibly very low accuracy (just better than chance). Low as a coin toss: so we've the 50% probability of providing the correct classification and 50% of returning an error.

x is a pattern to classify

$h_i(x) \in \{-1,+1\}$ are the weak classifiers

$\alpha_i \geq 0$ are the relative weights

$\sum_{i=1}^{M} \alpha_i$ is a normalization factor

$$H_M(x) = \frac{\sum_{i=1}^{M} \alpha_i h_i(\mathrm{x})}{\sum_{i=1}^{M} \alpha_i}$$

So we want to compose our strong classifier using weak learners. More than being a new classifier, the strong one can be expressed as a combination of the weak ones. The strong classifier is represented by the combination or the sum of the values returned by the weak classifiers weighted according to its relevance in the final decision (they return a value in the set {-1, +1} according to the presence or not of a certain feature).

In general -1 represents a negative outcome (non-face), while +1 is a positive detection (face). Once we carried out the weighted sum in order to limit ourself in the range (-1, +1), we've to divide this sum by the sum of weights.

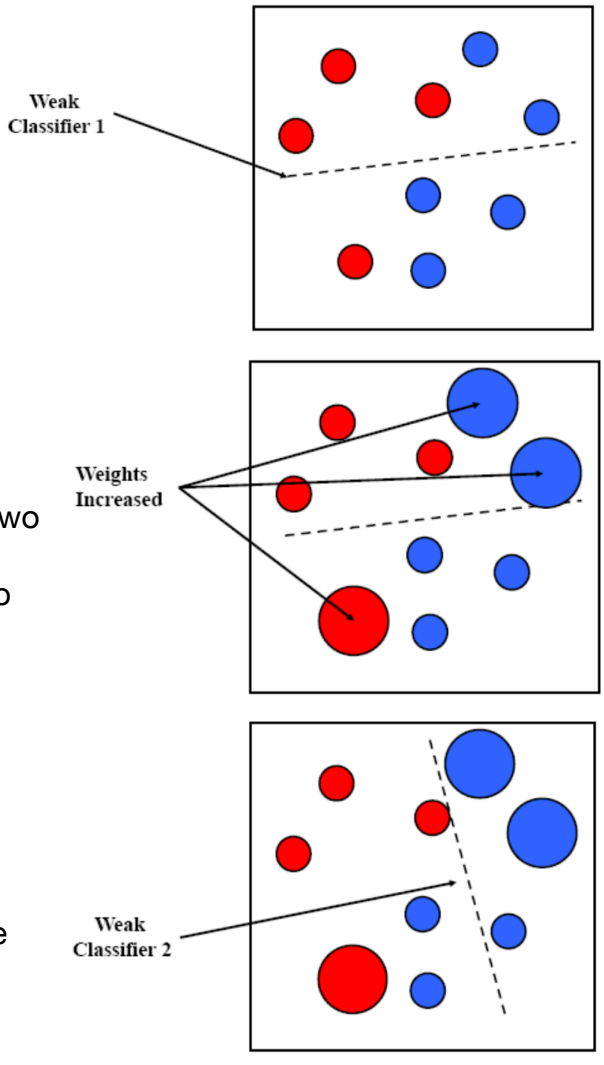
AdaBoost (**Adaptive Boost**) is a training technique that has the purpose of learning the best sequence of weak classifiers and their corresponding weights.

We've a set of training patterns $\{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$, where y_i is the **ground truth label** (because we've a ground truth during the training) that associates each pattern

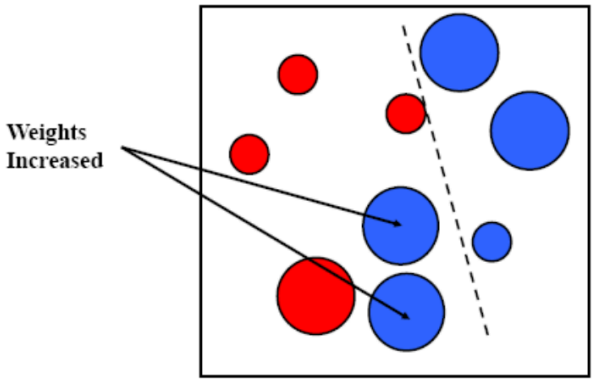
whether the presence of the object that we're looking for or the absence of such object.
So for each training pattern $x_1, x_2, ..., x_N$, $y_i = -1$ means that the pattern x_i doesn't correspond to a face, while $y_i = +1$ means that the pattern x_i contains the face.
In general we've each pattern x_i which belongs to a R^N dimensional space.
During learning we associate a **distribution of weights**: one weight for each pattern.
During learning a distribution of weights $[w_1, w_2, ..., w_N]$ associated with the training patterns $\{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$ is calculated and updated. This distribution that is associated to the training pattern, is computed and updated during learning.
At the beginning all samples have the same weight (e.g., 1), because at the beginning we've no information on what can the weight of each pattern be. Until we don't test the accuracy of the system over that pattern, we cannot say if it's a critical pattern or not.
After iteration m, a higher weight $w_i^{(m)}$ is assigned to the patterns that resulted more difficult to classify, so that in the next m+1-th iteration such patterns will receive greater attention. The most critical pattern will get the highest weight.
So it's a way to focus the attention of the system on a subset of the pattern, in particular to the most critical one.
Weight updating rules must be devised.
AdaBoost assumes to have a procedure for learning a weak classifier from a set of training patterns, given a certain distribution of weights $[w_i^{(m)}]$.

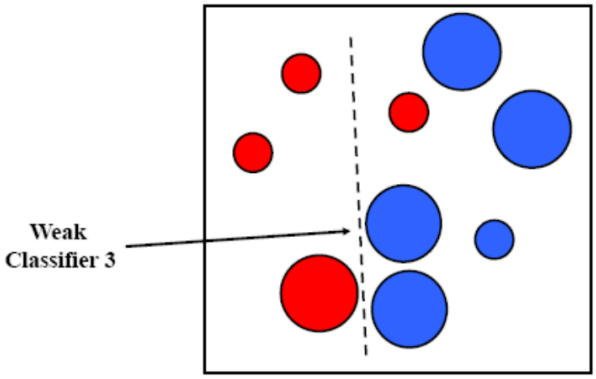
EXAMPLE

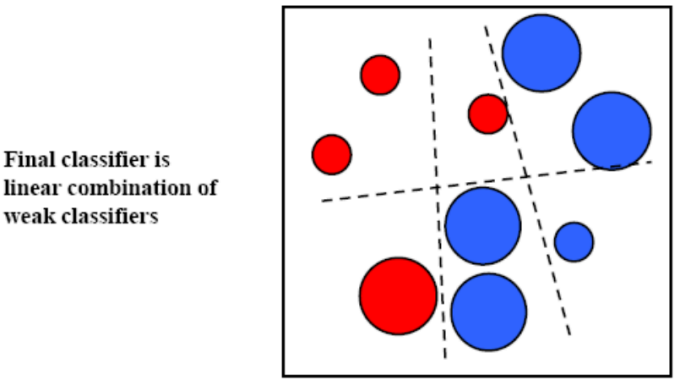
We've a two dimensional space of features, that means that each feature vector is represented by two features.
Our classes are represented by blue dots and red dots.
There is no linear classifier that is able to divide the two classes without any error.
Because we'll have always something that falls outside one class whatever is the position of the line.
This is **the limit of Weak Classifier**.
When we have used this linear classifier we've hypothesized the possible classification of the two classes.
Which are the errors in this example? The two blue dots upper the linear classifier and the red dot under the linear classifier. These samples have been misclassified.
In the next run, they will get the higher weight.
Which will be the task of the next classifier?
To concentrate on the patterns that have the higher weight. So we can divide with this weak classifier too.
So for each step, we're trying to find out the line that optimizes the sum of correctly classified weight with the respect to sum of incorrectly classified weight.
In any case we're trying to use a kind of optimization criterion: we're trying to minimize the weight of misclassified patterns.



Once we've increased the weight of these three patterns, we can imagine that this new classifier is the best classifier in order to divide the three blue balls from all the other ones. However, we've introduced new errors, because this time all the red balls are on the correct side, but the two blue dots have been incorrectly classified.
So we increase their weights for the next run.
In the next run it is as we start from the beginning in order to optimize the weight of incorrect classifications.
But this time, the patterns that have been created errors before, that have an higher weight.
Again, we've a new weak classifier.
The dots that gained an higher weight, maintain this higher weight.
Up to now, no one of the old errors have been repeated; otherwise some dots would have became even larger.
We've still one error, that is the red dot in the wrong side.
So in the next run, we will try to fix this error.
We've obtained the final combination of the three weak classifiers.
We've a combination of lines, so we don't have a linear classifier anymore.
If we apply these combination of lines, we will see that each region identified in the feature space contains only dots with the same colors.
So we've finished the process.

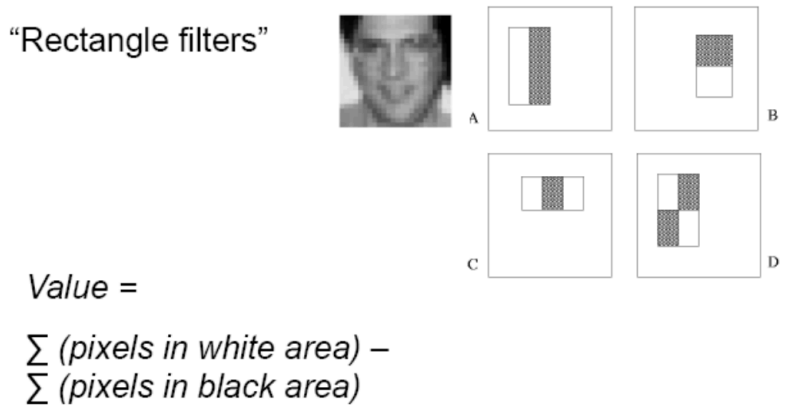




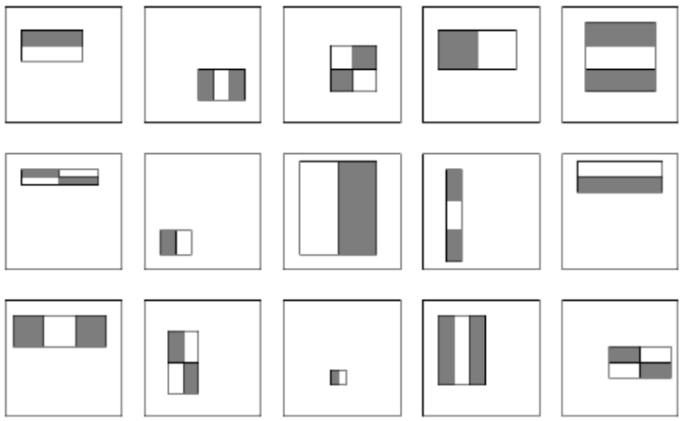


AdaBoost learning procedure for images

Which are the weak classifiers that we consider for Face Detection? As for images we can define weak learns based on simple (rectangular) features: **Haar features**.
They are **rectangular features**: we've a rectangle vertically divided in light and dark or horizontally divided in light and dark or with a mixed patterns.



Each feature is located in a subregion of a sub-window of the image (sliding window that we make slid on the image).
Features are applied by modifying their size, shape, and position in the sub-window.
In practice, for each sub-window we farther slide the feature within the sub-window.
What is the value returned by the system for these Haar features?
It is the sum of the pixels values that fall under the white area minus the sum of pixels values that fall under the black area.
So in practice these feature are like masks. We can establish a threshold for this difference so that it can somehow return a yes/no result according to the presence of this kind of pattern in the original image.
If we consider that each feature can slid over our window with different sizes, etc… we have a number of possible rectangular feature that is unfeasible.
For a 24x24 detection region, the number of possible rectangle features is ~180.000.
This kind of computation can be demanding for a computational point of view. So that it



would be useful to find a way to quickly compute the value of that difference. This is the computation of **Integral Images**.
At test time, it is impractical to evaluate the entire feature set.
We can use **AdaBoost** to select a small subset of all possible features to build a good classifier.

An **example** of an early stage in the **Haar cascade**. Each black and white patch represents a feature that the algorithm hunts for in the image.

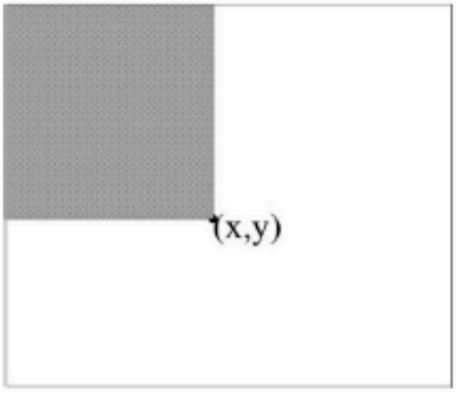
Let's return to the problem of **finding the best sequence of weak classifier**: each feature of that kind can be considered as a weak classifier, because if we take them one by one then they are not able to say us whether there is a face or not and we can say that each of those features once detected can only participate to the decision, but in itself it's more or less a coin toss.
So for each round of boosting evaluate each rectangle filter on each example. So for each pattern that we have in our training set both positive and negative we evaluate each rectangle filter and we choose the best threshold for each filter. The threshold that over the set of training examples is better suited to return a yes or no response with the respect to the presence of that pattern.
Then we select the best filter/threshold combination that is able to provide us the best final response with the respect to our problem and then we reweight the examples.
We reweight the examples because we run a classification using these and in particular we increasing the weight of the examples that had raised a misclassification.
The **computational complexity** of learning is tremendous: O(MNT) where M is the number of filters, N is the number of examples and T is the possible number of thresholds.

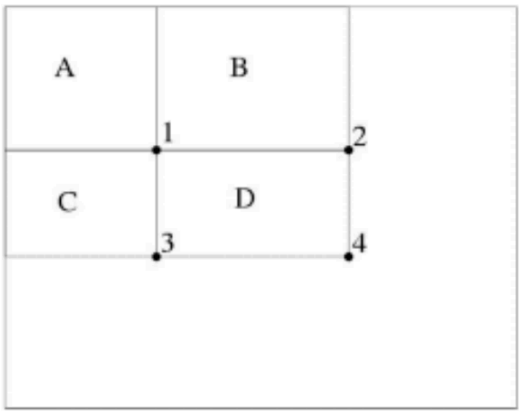
How to compute Haar features

Now we've the problem of computing the difference between pixels under the white area versus pixels under the dark area.
Integral Images are a solution for this problem. Rectangular features can be evaluated through integral images.
In practice, for each region the Integral Image is not but the sum of all the pixels where we've x' is lower than x and y' is lower than y.
The integral image in (x, y), denoted as II(x, y), is the sum of the values of pixels above and on the left of (x, y) computed:

$$II(x,y) = \sum_{x' \le x, y' \le y} I(x',y')$$

When we slide the window, there are pixels that will be taken again and again in the sum until they leave the window.
Using integral image it is possible to compute the sum of the values of pixels in any rectangle.
We pre-compute a certain number of sums in image rectangles.
If we take in this example the Integral Image in point 4 and the Integral Image in point 1 minus the Integral Image in point 2 and minus the Integral Image in point 3, we will have the rectangle D.
This thanks just pre-computed values.





Remember that Haar features are rectangular features that somehow describe vertical pattern or horizontal pattern or a mixed pattern.
Patterns work as kind of masks so that the computation of each feature is carried out by summing up the pixels that fall below the white area and subtracting the sum of pixels that fall below the black area.
This is mainly addressed to gray level images, because in that case is much easier to carried out this kind of computation; otherwise we could have an huge number of values either in RGB space or in other multi dimensional spaces.

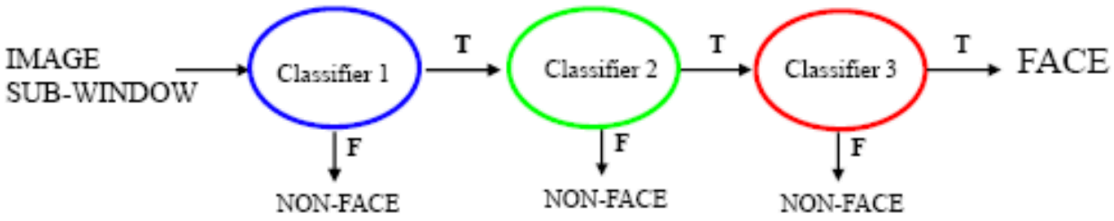
How is the computation of Integral Images connected to Haar features?
They are connected just by the need to compute the sum of the pixels. Since we don't change the original image, the values of the single pixels remain the same throughout the overall procedure.
What is changing during the slide of the window is the value of the difference between pixels, because both the sum of pixels below the white are and the sum of pixels below the black area may change during the sliding.
So it's important to optimize the computation of those sums.
The value of the same single pixel may enter many times in the computation, because everytime it is intercepted by the sliding pattern, it is considered again in one of the two sums (either in the white sum or in the black sum).

In general, whatever is the final shape of the region that we want to consider in our processing (whatever is the ratio between height and width), in the computation of an Integral Image what we do is to compute the sum of values of a certain rectangular pattern of pixels.
What we call Integral Image is the sum of the values of all pixels that fall in the rectangle with the x' dimension is lower than x and the y' is lower than y, where (x, y) is the point where we want to stop the computation.
So we have to sum up the values of those pixels within the rectangle, but we have to take into account that most of such pixels are already entered in some computation that has been carried out before while we were sliding the window.
The presence of Haar features represents a weak classifier. The simplest classifier is a decision tree with one node.
At the beginning all the samples are the same weights (samples are weighted according to the possibility to be misclassified: higher is the possibility, higher the weight in the next step).
Let's suppose we have built M-1 weak classifiers $\{h_m(x)|m=1,..,M-1\}$ and that we want to build $h_M(x)$.
We will stop when we will be satisfied with the accuracy of the obtained detector. When we're able to achieve a detection error that is small enough to be accepted.
Consider also that the index m is important because it's not just an index to identify a classifier from the other, but it's the order in which the classifiers must be applied.
What has this new classifier to do? This new classifier has to compare the value of a certain **feature z_{k^*}** with a fixed threshold **t_{k^*}** and assigns the values +1 or -1 accordingly to what happens applying that feature to that threshold.

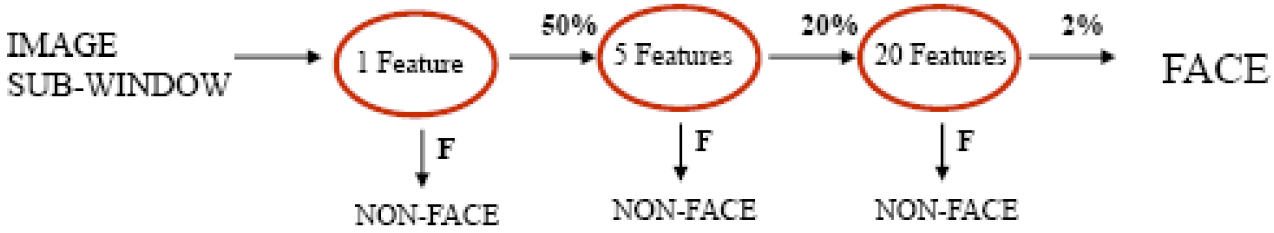
$$h_M(x) = +1 \text{ if } z_{k^*} > t_{k^*}$$
$$= \text{ -1 otherwise}$$

The threshold is what determines whether the difference between the pixels under the white area and the pixels under the black area is sufficient to determine if that pattern is present in the image or not.
In this approach each classifier provides a value that is either which class is 1 (that is presence of a face) or -1 (otherwise).
We're testing each feature with each possible threshold. So for each feature and for each threshold we mark the result of the classifier with +1 if the value of the feature is higher than the threshold and with -1 otherwise.
The choice of the classifier at step M $h_M(x)$ depends from the two parameters z_{k^*} and t_{k^*}. These two parameters are exactly what we've to learn, because that is what has to appear in the sequence after M-1 classifier and which is the threshold that must be applied at this point in order to provide a +1 or a -1 result.
The two parameters can be fixed based on the minimum classification error, that is: for each feature z_{k^*}, we choose the threshold value t_{k^*} which minimizes the classification error and the feature z_{k^*} which is choose for the current classifier is the one allowing to obtain the lower error.
So AdaBoost learns a sequence of weak classifier and combines them by minimizing the upper bound for the classification error.
For each feature z_k we choose the threshold value t_k which minimizes the classification error.
The feature z_{k^*} which is chosen for the current classifier is the one allowing to obtain the lower error.
AdaBoost learns a sequence of weak classifiers h_m and combines them in a robust classifier H_M, by minimizing the upper bound for the classification error of H_M.
We've to be careful in discarding image regions because once we've discarded a face region, this will not be taking into account anymore.

The ideal would be minimize the outcome for each step by discarding as mach as possible non-face regions and continuing with lowest number of possible true-image regions to be submitted to the next classifier. The problem is that once we discard a region, this will not be considered anymore and we will directly return a non-face partial result for that image region.
So we start with simple classifiers which reject many of the negative sub-windows; but the important thing is that it's able to detect almost all positive sub-windows. Because if we lose even a positive sub-window, the risk is that we cannot recover it anymore.
Positive results from the first classifier triggers the evaluation of a second (more complex) classifier, and so on.
A negative outcome at any point leads to the immediate rejection of the sub-window so that once we reject a certain sub-window in a certain point it will be labeled as non-face without continuing.
Only the image sub-windows that successfully run the overall chain will be classified as face.



How can we train this kind of cascade?
We adjust a weak learner threshold to minimize false negatives (as opposed to total classification error, because this is only provided at the end of the chain). So we try to minimize false negatives for each threshold.
Each classifier trained on false positives of previous stages.
A single-feature classifier can achieve 100% detection rate and about 50% false positive rate.
A five-feature classifier achieves 100% detection rate and 40% false positive rate (20% cumulative).
A 20-feature classifier achieve 100% detection rate with 10% false positive rate (2% cumulative).



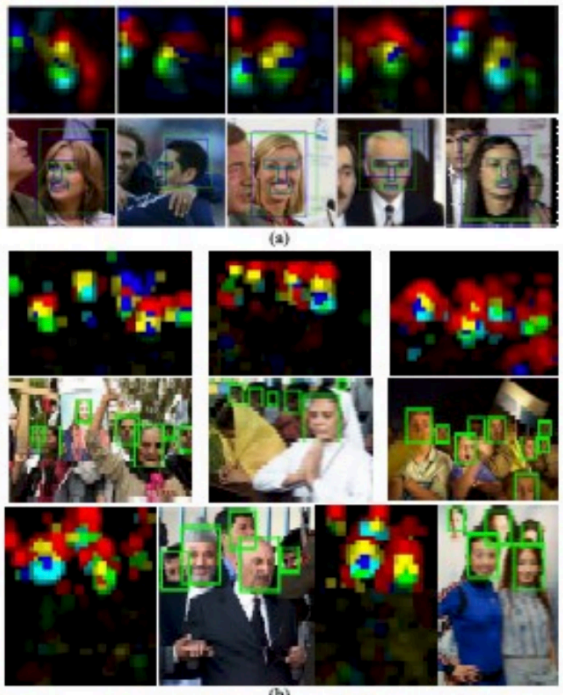
The localization of faces is done by analyzing consecutive sub-windows (overlapping) of the image as input and evaluating for each of them if it belongs to the class of faces.
The **locator of Viola and Jones** is one of the most robust and efficient in the state of the art: the training is very slow (it can take days) and the locating procedure is very efficient (real-time operation).

We've a number of reference samples that we've already processed in order to identify relevant landmarks so that when we look for a face in a new unsigned image. First of all we try to detect the landmarks and then we compare with the stored samples in order to see if there is a possible pose and expression whose landmarks can overlap with the new ones.

Another recent approach is based on **Deep Approach**.

The key of the approach is a new mechanism for scoring face likeness based on deep network responses on local facial parts.

The **part-level response maps** ('partness' map) generated by the deep network give a full image without prior face detection.



How is possible to evaluate Localization?

We've the **False Positives** that is the percentage of windows classified as faces that do not contain any face; we've the **Not Localized Faces**, the is the percentage of faces that have not been identified.

We can have an image with more faces within so that, in this case, we've to redefine this gross definition; because, for example, we can have an image with more faces where some of them were not localized but we can also have false positives within the same face. In this case, we've also to take into account which was the correct number of faces within a single images in order to better compute a reliable miss factor or a false positive factor.

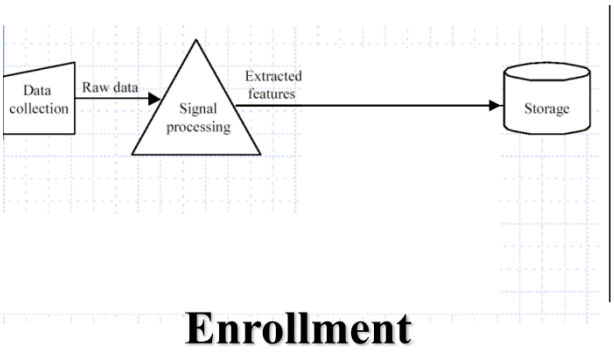
We can also further refine the evaluation of localization using the C-Error.

C-Error is not only based on the possible misclassification of the region of the image but also on the correct localization. The **Localization Error** is the Euclidean distance between the real center of the face and the one estimated by the system, normalized with respect to the sum of the axes of the ellipse containing the face.

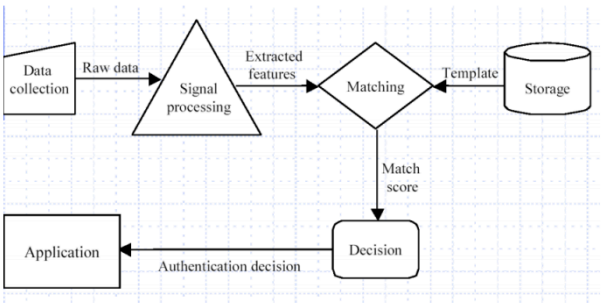
FACE RECOGNITION 2D

Structure of a face recognizer

In practice the structure of a face recognizer resemble the structure of any biometric recognizer.
We've an **enrollment phase**, where we associate a set of features to the identity of a subject. We've two separate phases of **raw data capture** and **feature extraction**.
Feature extraction entails a more or less complex procedure related to signal processing because we may consider that an image is 2D signal, but we also have one dimensional signals like the signal produced by an accelerometer. In any case we've a process of signal processing that ends with the extraction of features that are considered relevant for recognition.
Once we've extracted the features that can be of different kinds (for example the histogram of gray levels in the face can be a feature or for example the extraction of set of relevant geometric distances can make up a feature vector), the **extracted model**, that is also called **template**, is stored in a database or on a mobile holder (smart card).
In a database that is collected to tackle an identification problem, it's also necessary to associate an identity to a feature vector. So, in some cases, the so called **biometric template** not only contains the feature vector but also maybe some other information like the name, the address, the gender and so on and so forth, depending on the use that we want to do of this dataset.
The process for the enrollment can be performed in bunches (**batch enrollment**); this means that we can enroll groups of people in different times and also we can include in the enrollment more instances of the same biometric trait for the same subject in order to carry out a more robust multiple instance recognition.
The process stops with the feature extraction and storage; on the other hand, during the **testing**, that is during the recognition operation, the same same enrollment procedure is repeated on the so called **probe**, that is on the raw data that is submitted for recognition.
In general we can use in interchangeable way feature vector and template, but during the enrollment the template is enriched with other data, so the feature vector that is part of the template.
Then **the extracted model is searched** (template) is searched for in a database or on a mobile holder. At the end, there is a **matching criterion** that determines if the subject must be accepted or not.



Enrollment



Testing

We've **two main issues**:

- It is necessary to identify representative as well as discriminative features (we've already distinguished soft and hard biometric traits because for example the hair color is easy to extract and collect but it's not discriminative enough); at the same time the features that we extract must allow a well limited feature space that in some sense allows a real time operation. We've to take into account that it's very difficult to obtain a linear separations among classes, as well as convex classes.
  In general is very difficult to obtain a linear separation and that's why linear classifiers usually don't work and that's the reason why we apply cascade approaches in order to build non-linear hard strong classifier.
  We aimed to obtain the "**less complex**" representation as possible; not only the less complex but also the less demanding from the point of view of both computation and space requirements.
  In general is also useful and important to carry out a **normalization** of the feature vector that we have obtained because for example if we have an histogram of the gray levels in a face image this must be normalized with the respect to the total size; otherwise two histograms of two different face images of different sizes cannot be compared.
- Once we've chosen the features, it's also necessary to build the robust classifier able to **generalize** the training results.
  What is a good classifier? It's not only a classifier that is able to achieve good results on the training set, but it is most of all it's a classifier that is able to well generalize with the respect to never seen variations of an object.

POSSIBLE FACE REPRESENTATIONS

The most familiar one is to process an image that displaces a face. An image I (possibly a face image) can be considered as a **two-dimensional function** I(x, y) defined in the **Cartesian plane** (it can also be considered as matrix of values in two-dimensions), which associates a value (e.g., a grey level between 0 and 255 or a RGB color with each channel ranging from 0 to 255) to each position (x, y) in the plane. If we consider the image as in Cartesian plane, the origin of the image is on the bottom left; if we consider the image in the matrix representation the origin is on the upper left.
An image I is usually stored as a **matrix** of size w × h in which each element (single number or tuple) represents the value of a pixel.
Alternatively, an image can be represented as a **one-dimensional vector** of n = w × h pixels, for example by concatenating rows of the image one after the other.
More in general, an image I can be represented as a point in a multi-dimensional space.
A multi-dimensional space can be either the **Image Space**, so that the dimensions are two and we've a value for each pixel, or the **space spanned by the feature vector** that we extract from the image. We can associate a vector element for each feature that we extract from the image and once such elements are normalized then they can represent a kind of coordinates in a space. This is not equivalent to the familiar geometrical spaces that we usually know because in order to have something corresponding to the Cartesian spaces that we are familiar with, the axises over which the dimensions develop must be orthogonal; in general this is not true.
When we consider the overall Image Space (when we consider each image as w x h vector) it means first of all that we've normalized all images in order to have the same dimension and we've the so called problem of **curse of dimensionality**; we've many dimensions and each dimension corresponds to a position in the space (a cell in the image), so we've very scattered spaces and in a very far scattered space distances are very difficult to evaluate in a meaningful way because the higher is the dimensionality

(more scattered is the space), more the points may be far from each other even though they represent close elements. So, in this case, there are techniques for dimensionality reduction that are adopted.
When the dimensionality increases, the volume of the space increases so fast that the available data become sparse.
Sparsity is problematic for any method that requires statistical significance: in order to obtain a statistically sound and reliable result, the amount of data needed to support the result often grows exponentially with the dimensionality.
Data classification often relies on detecting areas where objects form groups with similar properties: in high dimensional data all objects appear to be sparse and somehow dissimilar, and this prevents data organization from being efficient.
Some **consequences**:

- **Machine learning** requires learning a possibly infinite distribution from a finite number of data samples; in a high-dimensional feature space with each feature having a number of possible values, an enormous amount of training data are required to ensure that there are several samples with each combination of values; with a fixed number of training samples, the predictive power reduces as the dimensionality increases (**Hughes effect**).
- When a measure such as a **Euclidean distance** is defined using many coordinates, there is little difference in the distances between different pairs of samples.
  This generates problems in nearest neighbor search and k-nearest neighbor classification.

On the other hand, we can also represent images in the **Feature Space**.
In the Feature Space we can apply the **Gabor filters** (they are two dimensional filters that are applied on the image and what is maintained are the coefficients of convolution of images with those filters). There are other possible operators: the **Discrete Cosine Transform** (**DCT**), the **Local Binary Pattern** (**LBP**) **Operator** and the **Fractal encoding** (e.g., Partitioned Iterated Function Systems – PIFS).
Dimensionality reduction methods such as **Principal Component Analysis** (**PCA**) aim at reducing high-dimensional data sets to lower-dimensional data without significant information loss, by identifying the most informative dimensions (the so called **relevant subspaces).**
Principal component analysis (**PCA**) is a statistical procedure that tries to carry out an orthogonal transformation that means that even though the original dimensions are not orthogonal, the resulting subspaces develop along **orthogonal dimensions**.
What is the meaning of orthogonal in this case? Orthogonal means uncorrelated. The more the values of different features are correlated to each others, the lower is the overall discriminative power of these highly correlated features just because they have the same trend. Once features have the same trend of values, they are poorly useful for discrimination. So orthogonal is a characteristic of uncorrelated features, so features that develop in a completely different way from each other.
While in correlation we can find a kind of **redundancy**, but also redundancy reduces the discriminated power.
So we try to avoid redundancy and we try to transform the initial set of values into linear uncorrelated values with no redundancy. These are called **principal components**.
The number of principal components that is extracted by this procedure is less than or equal to the number of original variables (in general is much lower than the number of original variables). In any case could be useful to transform the set of variables into an orthogonal space, but in general we also take the occasion to decrease the number of

dimensions. Once we've found the new space, we've to project the n-dimensional feature space onto the new k-dimensional subspace.
The transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to (i.e., uncorrelated with) the preceding components.
For an image space, the principal components are orthogonal when they are the eigenvectors of the covariance matrix, which is symmetric.
In signal processing it is also often referred to as the discrete **Karhunen–Loève transform** (**KLT**).
In practice, we take the linearized image; each linearized image becomes a column in a space; at the end we will have a matrix; we compute the transpose of that matrix and once we multiply the original matrix for its transpose we have the covariance matrix.
With PCA we're trying to solve the problem of **curse of dimensionality** because we're taking the overall image as a set of features corresponding to each and any pixels in the image.
In this case each pixel represents a dimension in the space and we're trying to reduce the dimensionality of what we're going to process to obtain a much smaller vector; but this doesn't mean that we're choosing a special region of face or that we're choosing subset of pixels.
We're carrying out a kind of transformation that will provide us a new lower dimensional space where each dimension cannot be considered as a subset or something derived from the original dimension.
This same procedure can also be exploited in order to reduce any other kind of feature space.
So we start from the **Training Set** that is represented by the linearized images so that each image becomes a column element of dimension one by n. So we take the first row; it becomes a column; then we attach a second peace of column representing the second row and so and so forth.
After this we compute the so called **Mean Vector**: we're taking all the m training vectors and summing up the corresponding components; then we compute the average. We obtain so a new vector that is of dimension 1 x n (so it is a column vector itself).
Once we have the training vectors and the mean vector, we subtract the mean vector from each training vector and then we compute the covariance matrix.
The covariance matrix is computed just multiplying each vector once it has been processed by subtracting the mean by the transpose of this obtained matrix. We can sum up the results on each corresponding component in order to obtain a new matrix whose dimension is n x m.
The optimal k is computed starting from a set of m n-dimensional samples making up the **training set TS** = $\{x_i \in R^n \mid i=1,\dots m\}$.
Given TS, we compute the **mean vector**: $\bar{x} = \frac{1}{m}\sum_{i=1}^{m} x_i$

We can now compute the **covariance matrix** for the vector set TS:

$$C = \frac{1}{m}\sum_{i=1}^{m}(x_i - \bar{x})(x_i - \bar{x})^T$$

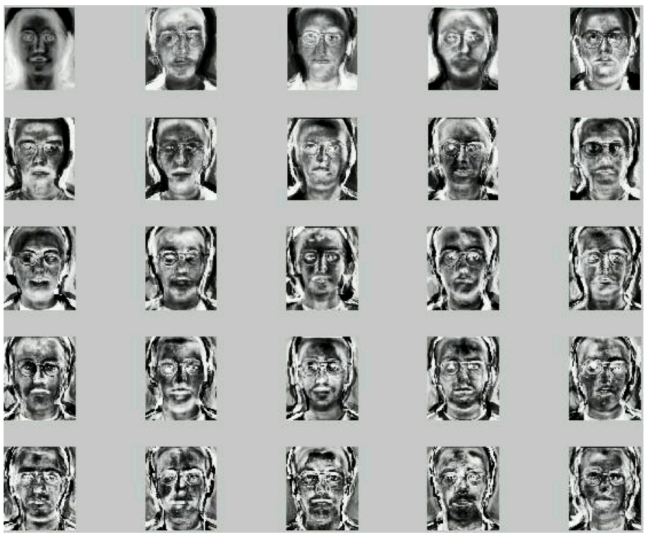
The size of the covariance matrix C is n × n, where n was the number of components in each original vector.

What we want to do is to extract by a space that has n dimensions a new space with a much lower dimensionality k, where in general we look for the smallest possible k by preserving the so called energy of the matrix.
Once we have the covariance matrix, we just proceed by looking eigenvector of such matrix.
How do we select the eigenvector? We consider the corresponding eigenvalues; each eigenvector corresponds to an eigenvalue; we order the eigenvalues in a decreasing order by noting which eigenvector they correspond; we choose the k eigenvectors corresponding to the highest k eigenvalues of the matrix.
The new k-dimensional space is defined by the projection matrix whose columns are the k **eigenvectors** of C corresponding to the k highest **eigenvalues** of C.
The eigenvector corresponding to the highest eigenvalue identifies the direction of maximum variation of data (so the direction with the **highest information**).
So the eigenvalues just represent the variances along eigenvectors.
Once we choose the highest eigenvalues these correspond to eigenvectors that can represent the basis where we have chosen the elements that present the highest variants among all the training images.
In this way, we automatically obtained the fact that these vectors are orthogonal to each other.
Sirovich and **Kirby** where the first to use Principal Component Analysis (PCA) for face recognition.
They demonstrated that any particular face can be efficiently **represented** along the **eigenpictures coordinate space**, and that can be approximately **reconstructed** by using just a small collection of eigenpictures and the corresponding projections ('coefficients') along each eigenpicture.
This means that we first take a set of training images; then we extract the new coordinates space (in practice the eigenpictures) that are able to represent the new space; each original picture, even the new coming ones, can be projected onto this new dimensional space.
A projection corresponds to finding out the coefficient representing the point from the original space onto the new one.
It's this coefficient by which we're able to reconstruct any image by using the eigenimages represents the new feature vector. That's why since we have k eigenpictures we can represent each original image like a k dimensional vector, where each element in this k dimensional vector is nothing but the coefficient of the projection of the original image onto the new space.
Starting from Sirovich and Kirby's approach to representation, **Turk** and **Pentland** realized that projections along eigenpictures can be used as classification features to recognize faces.
They developed a face recognition system that builds **eigenfaces**, corresponding to eigenvectors associated with the dominant eigenvalues of the covariance matrix built with linearized known faces (patterns).
The system recognizes particular faces by comparing their projections along the eigenfaces to those of the face images of the known individuals.
The eigenfaces define a feature space that drastically reduces the dimensionality of the original space, and face identification is carried out in this reduced space.
We can classify this approach in the class of **Holistic classification** because it takes into account the overall face without considering any special region as providing any kind of special information.

Example training set

This can be used as training set; so we take each of these images; we transform it in a column and then we compose a matrix that has as many rows as the full dimension of an image and as many columns as the number of training examples (25 in this case).
We take the columns of this matrix, we have summed them up and we have computed the average. This is what we intend by Mean Image from the training set.

The Mean Image represents what is more or less constant across the overall set.
Once we create the covariance matrix, we have a matrix that is n x n each column can be represented as an image itself. In this case we have the complete set of eigenfaces and this is what happens when we represent again the columns of the covariance matrix as they were face images; in the sense that we've again computed a square image by inverting the linearization problem.



The following step in PCA is to choose among these images which are the most representative in order to create a new feature space.

When we carry out the projection we take the transpose projection matrix and we multiply it again by the original vector.
Projection is performed by multiplying the **transposed projection matrix $\boldsymbol{\phi_k}^T$** by the original vector to which its mean was first subtracted.

$$Proj\colon \Re^n \to \Re^k$$

$$Proj(x) = {\varphi_k}^T(x - \bar{x})$$

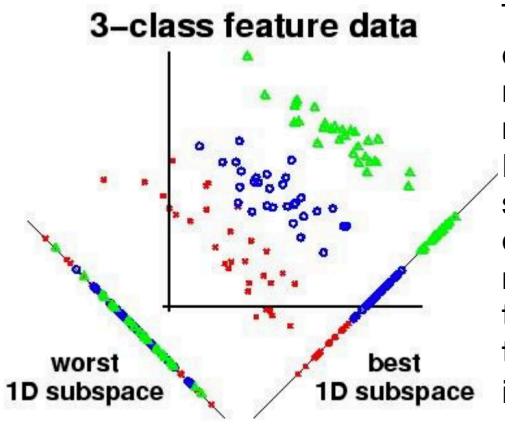
How is the projection matrix composed?
The projection matrix is composed by the k eigenvectors; they are the columns that we have taken by the n x n covariance matrix; such columns represent the transformation matrix. When we want to project a new image or even an image from the system gallery onto the new space, we can use the obtained matrix in order to map the image onto the new space.
The training set is important to create a meaningful new space because we must try to taking into account possible different characteristics of face images; but, if we play with the dimensions, we will see that in order to pass from space with dimensionality n to space with dimensionality k we have to carry out the multiplication in this way: we take the transpose matrix of the projection matrix (in practice the projection matrix is the transpose of the matrix that we obtain by taking the k eigenvectors extracted by the n x n matrix; we consider the transpose of this set of eigenvectors as the projection matrix) and multiply it with the linearized image after we have subtracted the mean.
The **projection matrix** only represented by which coefficients we have to multiply the original pixels in order to project them into the new space. So $\boldsymbol{\phi_k}$ is k x n. The **linearized image** that we want to project is n x 1. The **result** is a **k x 1 column matrix** that is exactly the column of coefficients that we have to use in order to carry out the projection.
So this k x 1 vector is nothing but the set of coefficients that must be used in order to project the original image x onto the new space.
Consider that we still maintain the average vector of the original images because we again subtract the most common features from the new incoming vector to before projecting it onto the new space.
The **Basic Idea** is that instead of comparing faces pixel by pixel, which provides an extremely unreliable result, we can rather compare such images once they are projected onto the k dimensional sub-space that has been built. **What do we compare?**
We compare the projection coefficients corresponding to each image. So the projection coefficients have no relationship anymore with the original appearance of the image.
So each gallery image is projected onto the KL subspace (that is what we obtain applying PCA) and the projection coefficients make up the feature vector. Once a probe image is submitted to the system, it's projected onto the new subspace as well to be recognized; the classification is based on a simplest nearest neighborhood criterion.
While we have to **identify** who is the subject in the image, we just compare its projection vector with the projection vectors of images in the gallery and, in case of identification, the face to be recognized is associated with the face of the gallery to which corresponds the minimum distance in the subspace.
This means that it’s not mandatory that each and any subject in the gallery participates in the training set. What is important is that the images used to build the new space are representative of the most relevant variations among subjects.
Given that once we’ve found the new subspace using the training set, then everything is projected onto it: both the gallery images and the probe images.

Which are the problems of PCA?
PCA is good for representation in the sense that it's very easy and reliable to map back the new feature vector onto the original image, but lacks true discriminative power. Eigenfaces separation (scattering) that is maximized depends not only on inter-class separation (truly useful for classification) but also on the intra-class differences. In the sense that the variance that is obtain along the dimensions represented by eigenfaces can be due either to inter-class but also to intra-class differences. But when we want to carry out recognition, we're much more interested in inter-class separation. So we're interested in that portion of variance that is due to inter-class separation rather than intra-class differences.
When there are significant PIE variations, PCA retains them and therefore the similarity in the space of the faces is not necessarily determined the identity of individuals. But may be determined by the fact that a face of a certain individual with a certain expression can be more similar to the face of another individual with the same expression instead of a face of the individual himself with a different expression.
To **solve** the above problems **Belhumer** proposed a variation of PCA: instead of using eigenfaces, use fisherfaces. **Fisherfaces** is an application of another standard technique that is **Fisher's linear discriminant** (**FLD**) and it's often mentioned in the context of **Linear Discriminant Analysis** (**LDA**).
The main difference between PCA and LDA is that the first one is unsupervised.
What does this mean? When we collect samples to be used to build the new feature space, we don't know which is the subject that has been pictured in in each picture. We just collect a number of representative enough samples from different subjects and we try to create a new feature space over which we map each gallery image and each incoming probe before comparing them. The comparison happens in a reduced space so that we have a less sparse distribution of points and also both clustering and comparison are more efficient and more accurate.
On the other hand, the main limit of PCA is that the variance over which the new projection relies can be either view to inter-class differences (and this is a good thing) but also to intra-class differences; so that, when we compute the eigenvectors, it can happen that some kind of variation belonging to the same subject in different situations is misclassified as the presence in a template of a different subject.
PCA is much more prone to intra-class misclassifications so it is more prone to false rejections in the sense that is is easier for PCA to be unable to recognize the same individual in a different situation.
The idea is to pass to **LDA that is a supervised method**; LDA is a supervised method because it takes into account not only the collections of samples but also a partitions of such samples into the different represented classes.

LINEAR DISCRIMINANT ANALYSIS (LDA)

LDA is a technique of linear supervised (samples in the training set are labeled) dimensionality reduction whose objective is to maximize the separation between the classes.
In PCA we've a kind of unsupervised training, in the sense that we're not interesting in the real identity of the training subjects; all the training images are collected together in the same training set.
Now, in LDA, we rather label the samples for each subject that participates in the training set. This kind of partitioning helps to better learn which are the elements in the new space that better refer to intra-personal variation with the respect to those that actually refer to inter-personal variations that are those that we're interested in.

Again we've an **optimization criterion**: in PCA was simply to take the eigenvectors with the highest eigenvalues; in this case we want to maintain the possibility to have images of the same person to fall in close position in the new space. But we also want to maximize the separation between the classes in this new reduced space.
The key to do this is to carry out a linear supervised dimensionality reduction, because we carry out a number of steps that at the beginning is similar to those for PCA but carried out on separated classes.



The space transformation is determined on the basis of an optimization criterion which has the aim of maximizing the separation between the classes in the reduced space.
In this figure we've an already good distribution of samples in the 2D space. We want to reduce from two dimensions to one dimension. It is not sufficient to reduce from two to one axis because if we consider the projection of points onto the X axis, we can see that the projections of points of differer classes get interlaced. So we also **rotate the axes** to maximize separation among classes.
Note that the projected values produce complete separation on the transformed axis, whereas there is overlap on both the original X and Y axes.
In order to be able to correctly identified the best subspace also from the point of view of classes division, what we do is to start again from a set of m n-dimensional samples ; but this time, these m n-dimensional samples are divided into the corresponded classes. In practice we assume that these samples belong to S different classes (in our case S different individuals) and each class has cardinality m_i.
We start from a set of m n-dimensional samples making up the training set:
TS = {$x_i \in R^n$ | i=1,...m}.
Differently from PCA, the training set is partitioned according to the known class labels of the s classes PTS = {P_1 , P_2 , ..., P_S} where each class P_i has cardinality m_i.
We seek to obtain a scalar y by projecting the samples x onto a line such that: $\mathbf{y = w^T x}$.
We may or not consider the same number of elements for the different classes. It is highly suggested to have the have the same number of elements for each class because it is a bad practice to have under represented classes.
On the other hand, the variance within the same class must be sufficient enough to avoid to use to much data that are useless.
In this example we have only **two classes**: P_1 and P_2. Without loss of generality, we can start from considering two classes P_1 and P_2 with respectively $m_1$and m_2 vectors.
Let us consider the **mean vectors** of each class in the two spaces:

$$\mu_i = \frac{1}{m_i}\sum_{j=1}^{m_i} x_j$$

and

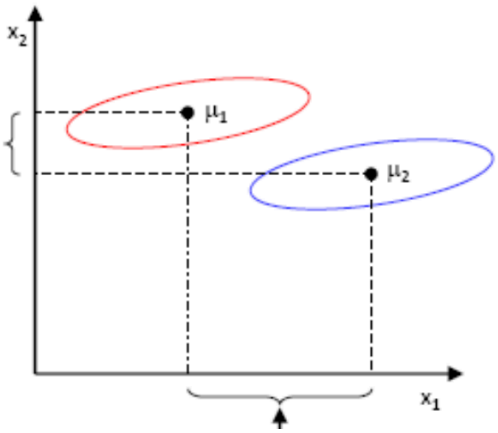
$$\widetilde{\mu_i} = \frac{1}{m_i}\sum_{j=1}^{m_i} y_j = \frac{1}{m_i}\sum_{j=1}^{m_i} w^T x_j = \mathrm{w}^T\mu_i$$

We could choose the distance between the projected means as the objective function (to maximize):

$$J(w) = |\widetilde{\mu_1} - \widetilde{\mu_2}| = |\mathrm{w}^T(\mu_1 - \mu_2)|$$

The distance between projected means is not a good measure because it doesn't take into account the standard deviation within classes, that is the scatter.

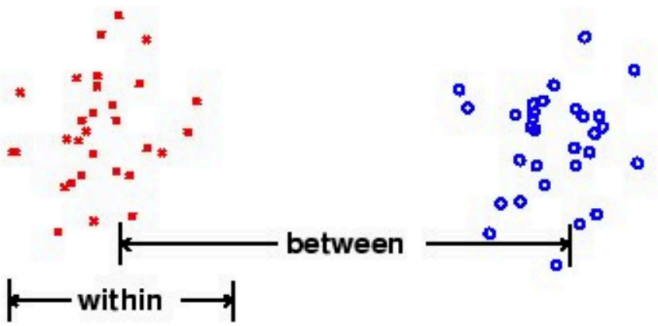
In this **example**, we've P_1 that is the set of points with centroid in μ_1 and P_2 that is the set of points with centroid in μ_2. If we just consider the distance between the two mean vectors we've a better class separability over the X_2 axis even though the highest distance among the average vectors can be found on the X_1 axis.



It's not sufficient to choose the dimension where the projected distance of average vectors is larger, because this may not necessarily held the better class separability.
So the pure distance among the projected average vectors is not a good criterion in order to find out the best projection of the original space.
The **starting step of LDA reduction process** resemble very closely the step that was applied in PCA, that is: we collect all the samples, but this time belonging to the same class, and then we compute the average vector. Consider that whenever we're able to represent our feature vectors as points in a space, when we compute the mean vector this mean vector can represent the centroid of the cluster of points collecting a certain class.
The centroid is relevant in order to determine how spread are the classes among them; but the problem is that the centroid is not sufficient because another element that must be taking into account is the spread of the samples in the same class around the centroid.

When can we obtain better results?

We start from two concepts that are the **Between-class variance** and the **Within-class variance**. We want to minimize the Within-class variance, in the sense that we want to have projected samples of the same class to be close to each closer to the new space; but we want also to maximize the Between-class variance, that is the variance among different classes.
Better results are obtained by maximizing the ratio of between-class variance to within-class variance starting from scattering matrices.



We start from **scattering matrices**: we can compute a **Within-class scatter matrix S_w** that indicates how the vectors of each class are scattered with respect to its center (the center is μ_i) and the **Between-class scatter matrix S_b** that indicates how the centers of the classes are scattered (i.e., how much the classes are scattered).
We want to **increase the between-class distance** and in this case the centroid could be an element to consider because it's more or less an indication of the spread between classes; but also we want to **reduce the within-class distance** so that we want have a compact representation of each class in the new space.

How can we obtain this?

First of all we can compute the so called Within-class scatter matrix that indicates how the vectors of each class are scattered across the class space with respect to the centroid; that is they provide the shape of the of border to see which direction this border is expanse more, so that it's easier in that direction to overlap with the other class.
Moreover we have the Between-class scatter matrix, that indicates how the centers of each class are scattered.

Again we start from a set of m n-dimensional samples making up the training set **TS = {$x_i \in R^n \mid i=1,\ldots m$}**.
Differently from PCA, the training set is partitioned according to the known class labels of the S classes PTS = {P_1 , P_2 , ..., P_S} where each class P_i has cardinality m_i. For each class P_i, we compute the mean vector (a «centroid» for each class similarly to PCA on the whole set) and the mean of means («centroid» of «centroids»).

$$\mu_i = \frac{1}{m_i}\sum_{j=1}^{m_i} x_j \qquad \mu_{TS} = \frac{1}{m}\sum_{i=1}^{S} m_i \mu_i$$

We compute the **covariance matrix** for the vector set P_i (for each class similarly to PCA on the whole set):

$$\mathrm{C_i} = \frac{1}{m_i}\sum_{j=1}^{m_i}(x_j - \mu_i)(x_j - \mu_i)^T$$

This time we've a covariance matrix for each class; so, for each class, we're looking for which are the higher variance dimensions.
The Within-class scatter matrix is exactly the weighted sum of the covariance matrices; weighted sum where the weight is the number of elements collected for each class.

$$S_W = \sum_{i=1}^{S} m_i \mathrm{C_i}$$

Then we've the Between-class scatter matrix that is built more or less in a similar way: we take the mean vector for each class, minus the mean vector of mean vector; again we compute a kind of covariance matrix and each multiplication is weighted again by the number of elements appearing in the class.
When we have only two classes, this can be simplified.

$$\mathrm{S_B} = \sum_{i=1}^{S} m_i(\mu_i - \mu_{TS})(\mu_i - \mu_{TS})^T$$

(for two classes it is also defined as $\mathrm{S_B} = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T$)

We can apply the same definitions in the new space, just using the projected versions of the original vectors. So that, since we want each new vector to be represented by the original vector multiplied by the transformation matrix, we can compute a sum of the squares of the differences between each vector and the mean value for the projection of that class.

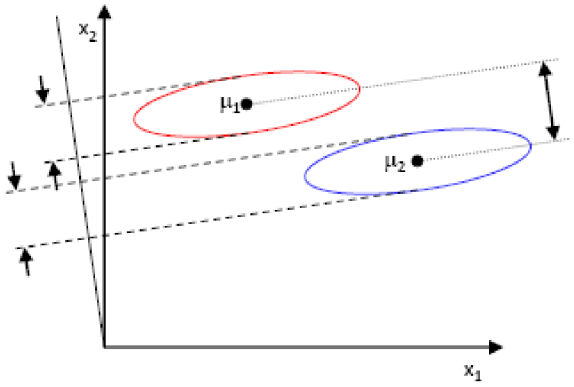
$$\tilde{s}_i^{\,2} = \sum_{y \in P_i}(y - \tilde{\mu}_i)^2$$

What we want to do is apply the **Fisher's solution**. Fisher suggested to maximize the difference between the means, so we still consider the difference between the projections of the two centroids, but this is normalized by a measure of the within-class scatter.
When we have only two classes , the Fisher linear discriminant is defined as the linear function w^Tx that maximizes the criterion function:

$$J(w) = \frac{|\tilde{\mu}_1 - \tilde{\mu}_2|^2}{\tilde{s}_1^{\,2} + \tilde{s}_2^{\,2}}$$

We are looking for a projection where we want maximize the numerator and at the same time we want to minimize the denominator.

Using the previous example, we notice that not only X_2 is the best axis (so we're not standing the fact that μ_1 and μ_2 are farther from each other on X_1) but we also slightly rotate that axis in order to farther increment the closeness of points in different classes while farther increasing the distance between the two centroids.
The difference between the projected means can be expressed in terms of the means in the original feature space:



$$(\tilde{\mu}_1 - \tilde{\mu}_2)^2 = (w^T\mu_1 - w^T\mu_2)^2 = w^T \underbrace{(\mu_1 - \mu_2)(\mu_1 - \mu_2)^T}_{S_B} w = w^T S_B w$$

Similarly, the scatter of the projection y can then be expressed as a function of the scatter matrix in feature space x:

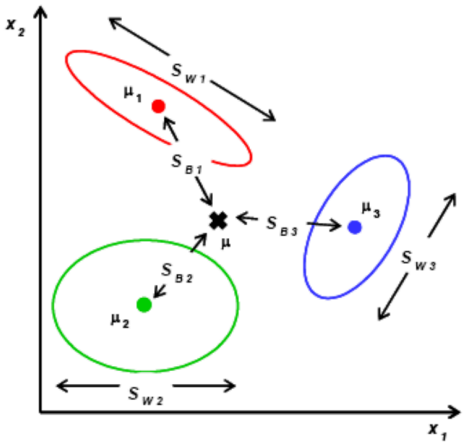
$$\tilde{s}_i^2 = \sum_{y\in\omega_i}(y - \tilde{\mu}_i)^2 = \sum_{x\in\omega_i}(w^T x - w^T\mu_i)^2 = \\ = \sum_{x\in\omega_i} w^T(x - \mu_i)(x - \mu_i)^T w = w^T S_i w$$

$$\tilde{s}_1^2 + \tilde{s}_2^2 = w^T S_W w$$

Therefore the Fisher criterion requires to maximize: $J(w) = \frac{w^T S_B w}{w^T S_W w}$

We seek the projection matrix W_* that maximizes this ratio.
It can be shown that the optimal projection matrix W_* is the one whose columns are the eigenvectors corresponding to the largest eigenvalues of the following generalized eigenvalue problem:

$$W^* = [w_1^*|w_2^*| \dots w_{C-1}^*] = \arg\max \frac{|W^T S_B W|}{|W^T S_W W|} \Rightarrow (S_B - \lambda_i S_W) w_i^* = 0$$



FEATURE SPACE

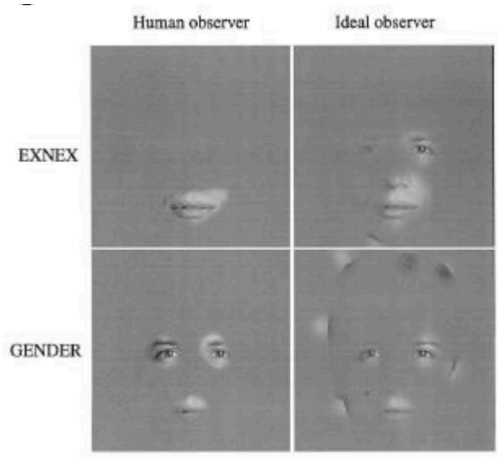
Up to now we've considered the Image Space, where each pixel in the original image represents a dimension in a very large feature space.
On the other hand, we can also extract different kinds of features from an image. In this case we assume that the number of features will be much lower than the number of pixels but it could be useful to reduce this space farther.
The relevant features of the image of a face can be extracted by applying image filters or transforms, different kinds of operators, each suited to detect particular properties.
When the dimensionality of the extracted feature vectors is high, they can undergo a process of dimensionality reduction using one of the previously described techniques.
It's interesting to take into account the comparison among our ability to recognize faces and the ability of an automatic system.

In this example, the challenge was to reproduce the ability of a human observer can easily distinguish the gender of the face image and also the sentiment expressed by the face.
The two images have the same hair dressing just to avoid any kind of queue that can be provided by the hair fashion that can be distinguish a male by a female and also they have the same pose.
They differ both for gender and also the kind of expression.


a.


b.

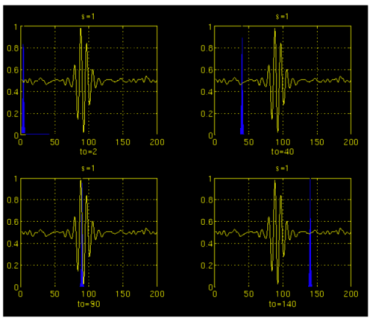
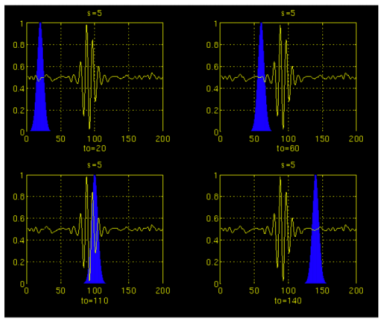
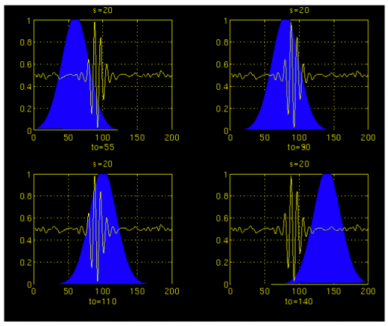
Which is the aim of this Bubbles experiment?
This figure illustrates diagnostic face information in an experiment for judging whether a face is expressive or not (EXNEX task), or its gender (GENDER task). The pictures are the outcome of Bubbles in the categorizations of the experiment on human (left column) and ideal observers (right column).
The system tried to detect which are the regions where the human observer focuses in order to distinguish an image and the regions where the so called "**ideal observer**" focuses.
This last one is a kind of deterministic processing in the sense that in this case the ideal observer just focuses on the regions with the highest variance.



Ideal observer in this experiment will capture all the regions of the image that have highest local variance between the considered categories (male vs. female, and neutral vs. expressive). This ideal considers the stimuli as images (not faces composed of eyes, a nose and a mouth, as humans do) and it might not necessarily be sensitive to the regions that humans find most useful (the diagnostic regions), but rather to the information that is mostly available in the data set for the task at hand.
The regions where the human observer focuses are not exactly the same where the ideal observer focuses.
The hypothesis is that this is the reason why is difficult to reproduce the advanced abilities of human in order to recognize faces.
Bubbles are random gaussian functions that are over imposed to the original picture. Everytime a certain pattern of bubbles produces the correct result either for the human or the ideal observer, it is added to a kind of access map and then a new random mask is presented. This mask is over imposed on the original image just to see whether the person or the ideal observer are able to classify either the gender or the emotion of the face.
The **result** is that the regions of attention are different between human and this kind of ideal observer.

WAVELET TRANSFORMS

In one dimension, the **Fourier transform** is used to approximate a kind of periodic and continuous signal with a number of sinusoids, so that Fourier transform consists in computing the coefficient to apply to each sinusoid in order to obtain the original signal. The coefficients of the different sinusoids can also be used as feature vector for that signal.
When we consider the Fourier transform we've a situation where we've a sharp distinction between the frequency domain and the time domain. Any information that is available in one stage, it's not available in the other.
In particular we have no time informations when we apply a Fourier transform. We're able to compute which are the frequencies that are presence in the signal, but not to where. So they can be applied to the so called stationary signals, so that we're not much more interested in when that frequency is produced in the signal, but just in its pure presence in the signal.
What is the solution?
The solution is to use a more advanced **wavelet transform** that doesn't use the Fourier sinusoids, the so called **Mather Wavelet**. It provides a kind of shape (a prototype) for the wavelets that we convolve with the signal so that starting from the Mather Wavelet with different period, we're able not only to obtain the frequencies, but also the temporal dislocation of each frequency.
In practice, if we consider the spectra of two entirely different signals processed by Fourier transform, there are thesis where they may seam similar just because we've only the information about the presence of that frequencies; so, if the same frequencies are present in the two signals but in two different times, they may appear as similar.
This is why Fourier transform can be used with the not stationary signals only if we're interested in the presence of a certain frequency.
In general, if we consider an image as a 2D signal, we're not only interested in the frequency of a certain component, for example the grey level, but we're also interested in exactly knowing where that grey level appears with a higher relevance.
Wavelet transform is capable of providing the time and frequency information simultaneously, hence giving a time-frequency representation of the signal.
A "**mother wavelet**" is shifted in time-domain, and the process is repeated with different scales (inversely proportional to frequencies).
The term wavelet means a small wave. The **smallness** refers to the condition that this (window) function is of finite length (it has a compact support, which means that it vanishes outside of a finite interval). The **wave** refers to the condition that this function is oscillatory. The term **mother** implies that the functions with different region of support that are used in the transformation process are derived from one main function, or the mother wavelet. In other words, the mother wavelet is a prototype for generating the other window functions.
The yellow line is the origin signal, while the blue wave represents the wavelet.

We can change the period of the wavelet, that is the frequency (how many times the wave crosses the 0 axis in the unit period of time).
If we **increase the frequency** we have a narrow wavelet, because the increase of frequency means that they cross the 0 more times. The narrower is the wavelet, the smaller is the detail of the underline signal that is able to detect.
When we convolve a wavelet (provides a higher value when the underline signal corresponds to the kernel that we're convolving) with a continues signal or a kernel with an image, we get a higher value when pattern that is present in the wavelet (in this case the frequency) is also presents in the underline image.
When we **decrease the frequency**, we've wider waves and we're able to catch the bigger details and in the case of wavelets we can capture the lowest frequency presents in the original signal.

$$CWT_x^{\psi}(\tau, s) = \Psi_x^{\psi}(\tau, s) = \int x(t).\psi_{\tau,s}^*(t)\, dt$$

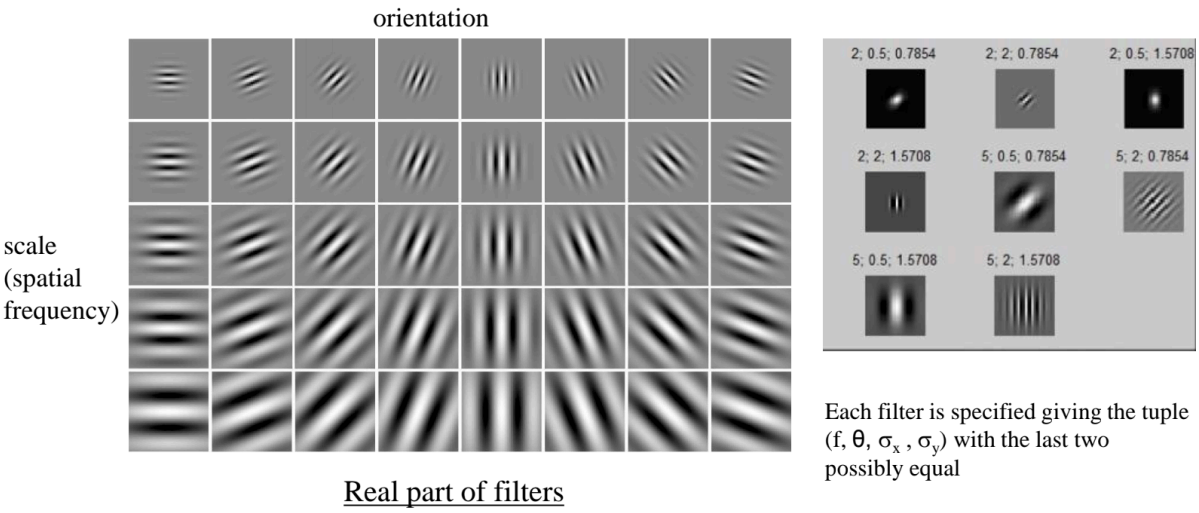
$f^*(x)$ is the complex coniugate of complex function $f(x)$

$$\psi_{\tau,s} = \frac{1}{\sqrt{s}}\psi\left(\frac{t-\tau}{s}\right)$$

Wavelet algorithms process data at different scales or resolutions. If we look at a signal with a large “window”, we would notice gross features. Similarly, if we look at a signal with a small “window”, we would notice small features. The result in wavelet analysis is to see “both the forest and the trees”.
Temporal analysis is performed with a contracted, high-frequency version of the prototype wavelet, while frequency analysis is performed with a dilated, low-frequency version of the same wavelet.
A 2 dimensional wavelet can be expressed as a kernel, that is sliding over the image and for each sliding step there is convolution value that is computed in order to transform the image.
In two dimensions:

$$\psi_\theta(b_x, b_y, x, y, x_0, y_0) = \frac{1}{\sqrt{b_x b_y}}\psi_\theta\left(\frac{x-x_0}{b_x} + \frac{y-y_0}{b_y}\right)$$

Except for spatial shifting, Gabor filters are compatible with this expression.
A kind of wavelet that is exploit with images is the so called **Gabor filters**.
The Gabor filters is a filter that is basically used for **edge detection**. The frequency and the orientation that somehow characterize a Gabor filter are the same that can be found in the visual cortex of mammalian brains.
So in practice there is an attempt to reproduce the elementary processes that allow humans to carry out some perceptual tasks.
Thus, image analysis with Gabor filters is thought to be similar to perception in the human visual system.
A Gabor filter works as a bandpass filter (is a filter that captures only frequencies in a certain interval and cuts everything below or above the interval of frequency) for the local spatial frequency distribution, achieving an optimal resolution in both spatial and frequency domains.
The 2D Gabor filter $\psi_{f,\theta}(x, y)$ can be represented as a **complex sinusoidal signal** modulated by a Gaussian kernel function.
What characterize a Gaussian function are the standard deviation and orientation.
The filter has a real and an imaginary component representing orthogonal directions. The two components may be formed into a complex number or used individually.



Real part of filters

Each filter is specified giving the tuple ($f, \theta, \sigma_x, \sigma_y$) with the last two possibly equal

This is a bank of filters. Each of this image is a filter that we convolve with the original image in order to capture different kinds of pattern.
A feature vector is obtained by convolution of an image with a **Gabor filter bank** (different orientations and different scales).
In the first row, what is change is the orientation of the filter; on the other hand we can change the scale of the filter that represents the spatial frequency.
Combining different orientations and scale, it's possible to obtain a huge bank of filters.
What happens with images?
We compute the convolution of these filters with different regions of the image; we slide these convolution filters over the image on we compute the response in order to detect the presence of patterns of this kind.

2D Gabor functions enhance edge contours. This corresponds to enhancing eye, mouth, and nose edges, and also moles, dimples, scars, etc.
If the convolution is performed on all the pixel of the image and we take the overall result, dimensionality is hight (size of the image). In alternative it can be performed on a regular grid of points, or on salient face regions only.
Going from top to bottom, we obtain details of different size. This kind of process is **computationally expensive** because we've to carry out the convolution as many times as is the size of the cardinality of the filter bank.
Moreover, there is another problem: the result of each convolution is itself an image of the same size of the original one, so that this can be considered as a feature based representation of the original image.
However we've increased the complexity of our problem. A possible solution that can be applied in general to reduce from the beginning the dimensionality of the features that we

obtain, we can overlap a grid on the original image and compute the convolution only in relevant points in this grid.

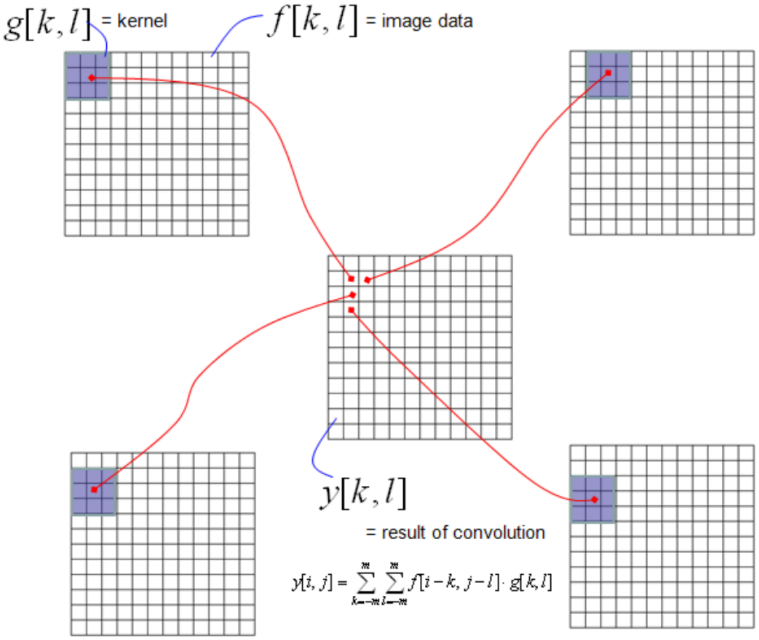
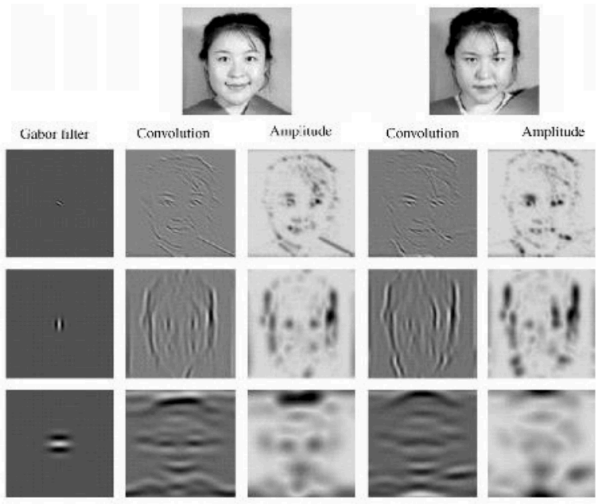
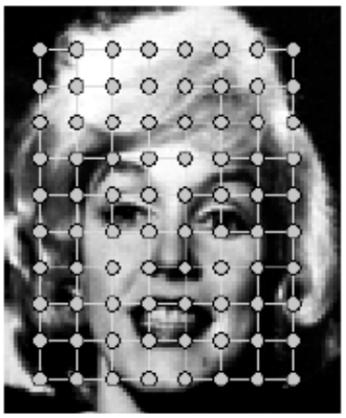


Image Convolution is not a simple multiplication of matrices.
In this case we have a 3x3 kernel that is slided over the underline image and the result of each point goes to the point in the mean image that corresponds to the center of the kernel.
Each time we slid, the result of convolution goes in the new image in the point corresponding to the kernel.
In this example we've two images of the same person with different expressions.
We have three different filters; the "amplitude" is obtained by considering the combination of the real and the imaginary path.

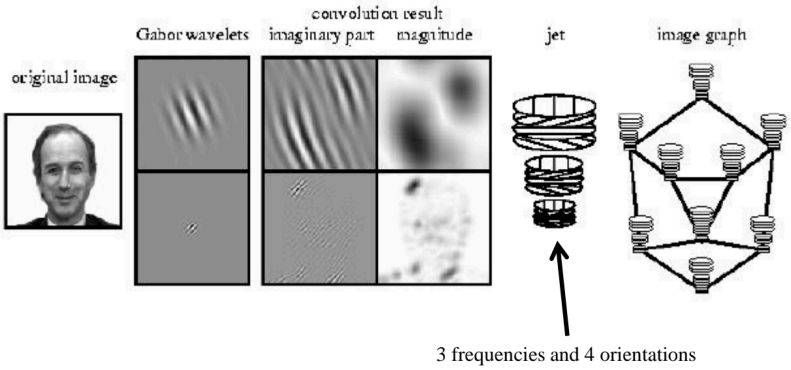


We can see what happens when we apply the same filters to images of the same person.
Applying these filters we can't distinguish the expression and we obtain much more similar results.
The following example is an example of how we can reduce the complexity by carrying out the convolution only with the fixed grid.
The **grid defined «by hand»** such that vertices are close to important facial features such as the eyes. We decide to exploit only in those points to compute the wavelet response.



This «by hand» has two **drawbacks**: first of all, we want make the results comparable but the grid decided «by hand» could catch the relevant points in one image and miss the same points in an other image; moreover we can miss some relevant points that could be useful for recognition.
The other option is to select the **high energy points** (points where the response of the Gabor wavelet is higher) and then use only this points as feature vectors.
The computation is carried out again over the overall image.
An application of Gabor filter is the **Elastic Bunch Graph Matching (EBGM)**.
"Bunch" represents a set of objects that were produced during the application of the approach.

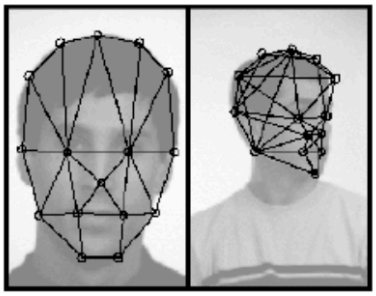
The basic object representation is a labeled graph, defined as bunch graph, one for each pose, which collect class-specific information. A graph for each possible pose.
Edges from one node to the other are labeled with distance information and nodes are labeled (contained) with Gabor wavelet responses locally collected in jets. So not only a single response to a Gabor wavelet but a jet that collect responses of different filters in the same region of the face.
Bunch graphs are stacks of a moderate number of different faces, jet-sampled in an appropriate set of **fiducial points** (are relevant landmarks in a face like eyes, mouth, contour, etc.).
A **jet** describes a small patch of grey values in an image I around a given pixel p = (x; y). It is based on a wavelet transform (5 frequencies × 8 orientations) providing a set of 40 coefficients for one image point.

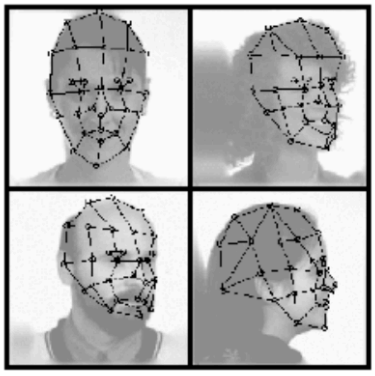


A sparse collection of such jets together with some information about their relative location constitutes an image graph, used to represent an object, such as a face.
Then we have a Bunch Graph that is created by collecting for each fiducial point all the jets for the 70 people that were included in the model.
A bunch graph is created in **two stages**:

- Its qualitative structure as a graph (a set of nodes plus edges) as well as the assignment of corresponding labels (jets and distances) for one initial image is designer-provided
- The bulk of the bunch graph is extracted semi-automatically from sample images by matching the embryonic bunch graph to them, less and less often intervening to correct incorrectly identified fiducial points.

Individual faces: a labeled graph G representing a face consists of N nodes on a set of fiducial points (e.g. the pupils, the corners of the mouth, the tip of the nose, the top and bottom of the ears, etc.) and E edges between them. The nodes are labeled with jets and the edges are labeled with distances. This face image graph is object-adapted, since the nodes are selected from face-specific points (fiducial points). Graphs for different head pose differ in geometry and local features. Although the fiducial points refer to corresponding object locations, some may be occluded, and jets as well as distances vary due to rotation in depth.


grids for face finding

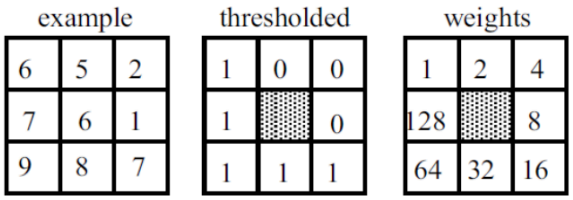

grids for face recognition

To be able to compare graphs for different poses, pointers are manually defined to associate corresponding nodes in the different graphs.
To find decimal points in new faces, one needs a general representation.

The general representation should cover a wide range of possible variations in the appearance of faces, such as differently shaped eyes, mouths, or noses, different types of beards, variations due to sex, age, race, etc.
A representative set of individual model graphs is combined into a stack-like structure, called a **face bunch graph** (**FBG**). Each model has the same grid structure and the nodes refer to identical fiducial points.
A **set of jets** referring to one fiducial point is called a **bunch**.

LOCAL BINARY PATTERN (LBP)

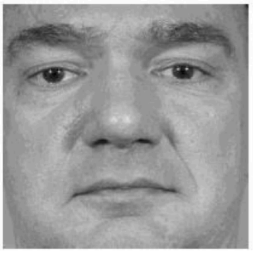
LBP is a texture-based operator. In image processing a **texture** is whatever is somehow composed by a recurrent pattern. In general texture-based operators aim to find out which are the recurring patterns that make up a region possibly to distinguish regions presenting in a certain texture from regions presenting in a different one.
LBP can be used in **two ways**: either as an **operator** or also as a **pure texture-operator** in order to detect special patterns that are produced in the image when this is recaptured from a video or a printed photo is shown instead of a real face.
LBP **works pixel by pixel**. We have a sliding window, but in this case we don't use a kernel like in integral images or convolution. The image itself provides all the necessary informations in order to carry out the processing that we want to obtain.
Assume to have an image of a certain size we can apply LBP at different resolutions like wavelets, so the typical resolution is to exploit a neighborhood of size 3×3 for each pixel. The neighborhoods are conventionally assumed to be of odd size in order to be able to center each time the window over a certain pixel in the original image.
For each pixel, the basic version of the operator considers its neighborhood of size 3×3 and assigns to each pixel therein a binary value (0 or 1).
For each such window we use the value of the central pixel as it was a **threshold**.
In LBP computation, each pixel once it becomes the central pixel in a LBP window, it becomes the local threshold. It's a kind of extremely **adaptive thresholding procedure**. So for each 3x3 window, the threshold value is the value of the central pixel.



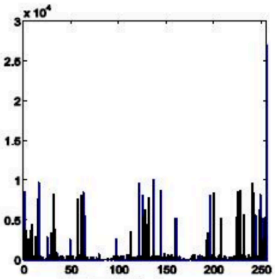
Pattern = **11110001**
LBP = 1 + 16 +32 + 64 + 128 = **241**
C = (6+7+8+9+7)/5 - (5+2+1)/3 = **4.7**

The result of computation is assigned in the new LBP image that we're building to such central pixel.
How do we compute the new value for the central pixel to map into the new image?
We consider the neighborhoods of the pixel. For each neighborhood we compare it with the central pixel. When the value of the neighborhood is equal or higher than the value of the central pixel, we consider in its place a 1. Otherwise we consider 0.
How can we interpret this set of binary values?
We can interpret them as the binary digits of a binary number. So we can translate this pattern of 1 and 0 into a binary number. In doing this we have also to decide a starting point: **where does the binary number start?** We have a kind of circular pattern.
According to the point from which we start coding the binary digit we can have a different value.
Let's say that we start from the first one; the weight assigned to the first pixel is 1, because it's 2 power 0 (it's the first digit on the right of. the binary number); then we assign weight 2, that is 2 power 1; and so on and so forth. In this way we can compose the binary number.

The first element that provides a value to compute the final digital number corresponding to the pattern in that cell, in this case is the bottom right one.
We can conclude that in the new image the central pixel that we've centered the window on will assume the value 241.
Every time we use a 3x3 window or we use only eight pixels within the LBP window, what we can obtain for each pixel is a value between 0 and 255.
The resulting image will be a grey level image that can be interpreted as an image where contrasts are especially stressed.
We're still processing the image to obtain a new image. We're transforming an image into another one that can be either used itself as a feature vector for the original image if we adopt an image space representation or it can be used to extract features that instead of be extracted from the original image, they will be extracted from the LBP image.


Input image


LBP image

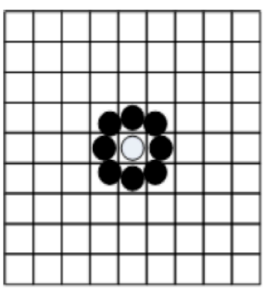


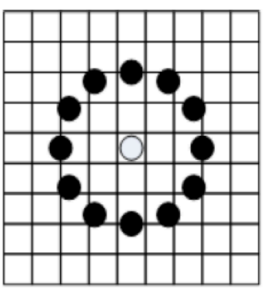
LBP histogram

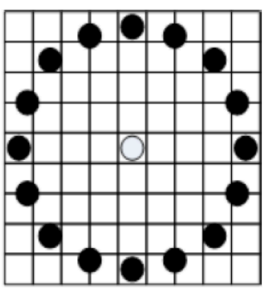
Another information that we may extract is also a kind of contrast value that we can assign to each pixel in a third kind image that is called "**contrast image**".
Furthermore, a kind of **contrast measure** referred to the central pixel can be computed, subtracting the average value of neighbors with a higher or equal value from the average vale of neighbors with a lower value.
LBPH is one of the most popular ways to exploit LBP processing is not much to exploit the image but to compute the histogram of values that can be extracted from the grey levels of the LBP image. We don't extract the histogram from the original image but from the LBP image.
During the years, the original method has been modified to handle interval of different radius.

Which are the main variations of LBP?

The first variation is the use of a larger window, so to handle intervals of different radius. In practice, the 3x3 window has a radius of 1 (we take pixels at distance 1 from the central one); the second parameter is the number of neighbors that we consider, so in the case in the radius of 1 the maximum number of neighbors that we can consider is 8 (but we may consider less than 8, for example only those in the main directions).
The next step is to increase the radius. We must consider to increase the radius in a way to have a central pixel. In the second example we have a 7x7 window with the radius of 2.5. In this case we can consider more than 8 neighbors, but we miss the representation as an image because there is no image with a value for a pixel larger than 255. But in any case we can compute the histogram, because it's still a feature of that image that we can extract applying exactly the same kind of computations with 12 neighbors.


P = 8, R = 1.0


P = 12, R = 2.5

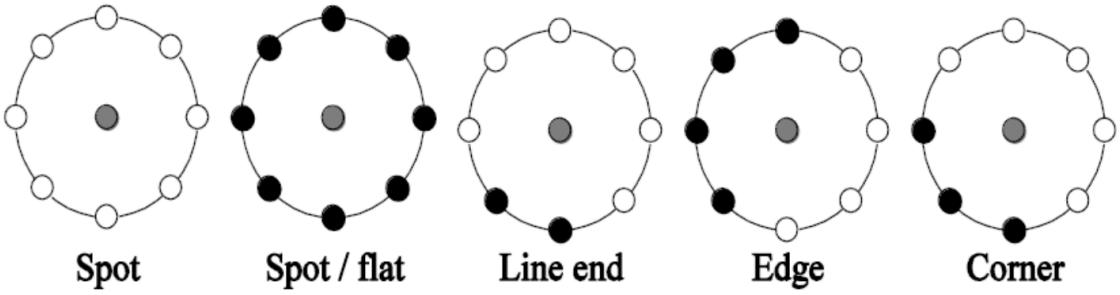

P = 16, R = 4.0

The number of neighbors can catch more detailed information, but the resolution of the details that we're able to capture depends on the width of the mother window that we exploit to compute the

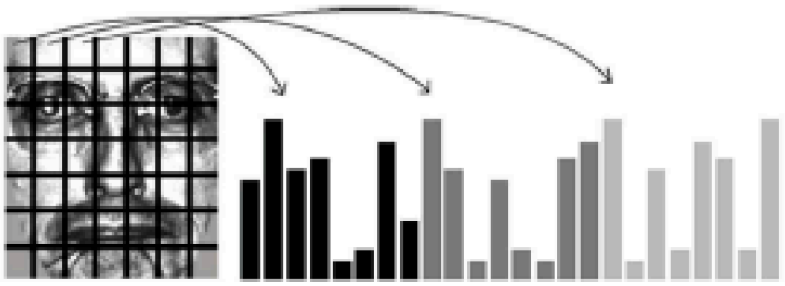
LBP values. So with small windows we can detect fine details; with larger windows we can detect larger details (like with wavelets).
The value for the LBP code of a pixel is provided by the binary coding of the result of the pixel by pixel comparison with the central one.

$$LBP_{P,R} = \sum_{p=0}^{P-1} s(g_p - g_c)2^p \qquad s(x) = \begin{cases} 1, if\ x\ \geq\ 0; \\ 0, otherwise. \end{cases}$$

Not all the LBP codes have the same relevant for the final coding of the image, so we can consider only the so called uniform patterns.
Uniform patterns are useful to save memory, because once we've a number of neighbors that is P, we only have P x (P-1) + 2 bins (+2 means the case of Spot and Spot/Flat) for a total of P^2 bins. We consider as uniform all patterns where there are a maximum of two changes from 0 to 1.
Uniform patterns are relevant since they identify significant structures.



The feature vector associated with an image is a **histogram** (possibly normalized) calculated as follows: the image is partitioned into sub-windows, e.g., according to a square grid of k×k elements; we can compute the LBP histogram for each element in the grid in which each bin is associated with a specific pattern (the number of bins depend of the LBP type); for obtain the final feature vector we must chain as many histogram as the number of cells in the grid.



What happens if we change the starting point for the coding?
Especially when the picture is rotated, so we have the same face but rotated with a certain angle, we may obtain different values just because the argument of the function is rotated. In this case we can apply the rotation invariant form of LBP, that in practice for each pattern of 0 and 1 that we obtain, always chooses the lowest decimal number that we can obtain from that window.

Global (holistic) appearance methods: based on the whole appearance of face.
For example:

- Principal Component Analysis (**PCA**), Linear Discriminant Analysis (**LDA**), Independent Component Analysis (**ICA**): linear methods also used for dimensionality reduction. All of these linear methods represent a face as a linear combination of basis vectors or as an addition of subcomponents.
- Some Artificial Intelligence (**AI**) approaches: neural Networks.
- **Sparse representations**: the «basis» is the whole set of training images. In sparse representations each and any linearized face image is part of the basis without any kind of reduction and so the space that we obtain is extremely sparse. But when we map a face into this extremely sparse representation space, the most close basis vector will be those belonging to the same person.

Which are the advantages and the disadvantages of global methods?

- They do not destroy any of the information in the images (because whatever is in the original image is somehow preserved even though that information is not interesting) by concentrating on only limited regions or points of interest. ADVANTAGE
- We've in parallel the disadvantage of this approach that each and any pixel is considered as relevant when it's not necessarily. DISADVANTAGE
- Several of these algorithms are flexible enough to have been modified and/or enhanced in order to address and compensate for PIE variations, and dimensions. ADVANTAGE
- These techniques are computationally expensive. DISADVANTAGE
- They require a high degree of correlation between the test and training images. DISADVANTAGE
- They do not perform effectively under large variations in PIE as well as scale, etc. DISADVANTAGE

On the other hand, we have the **Local or feature-based methods**: based on relevant points or on local characteristics of single zones, possibly anatomically significant. For example:

- Elastic Bunch Graph Matching (**EBGM**)
- Local Binary Patterns (**LBP**): method used for computing textures

Which are the advantages and the disadvantages of Local or feature-based methods?

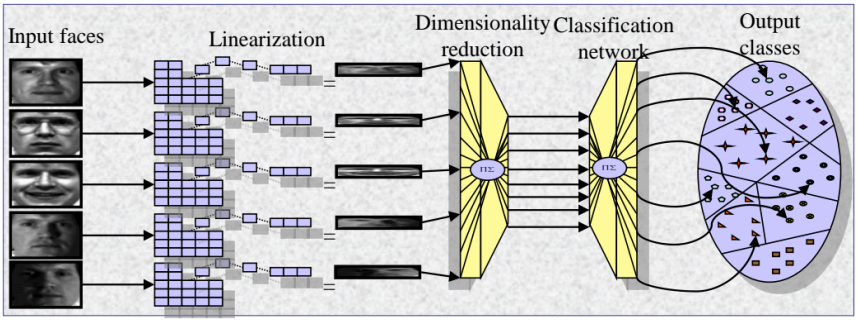
- The extraction of the feature points precedes the analysis done for matching the image to that of a known individual, therefore such methods are relatively robust to position variations in the input image. ADVANTAGE
- Feature-based schemes can be made invariant to size (because it's only necessary normalize the values of feature vectors according the image size), orientation and/or lighting. ADVANTAGE
- They allow a compact representation of the face images (that allow a much quicker comparison) and high speed matching. ADVANTAGE
- The implementer of any of these techniques has to make arbitrary decisions about which features are important. DISADVANTAGE
- If the feature set lacks discrimination ability, no amount of subsequent processing can compensate for that intrinsic deficiency. DISADVANTAGE

Which are the advantages and the disadvantages of the Eigenfaces?

- The identification phase is fast. ADVANTAGE
- If eigenvectors are preserved it is possible to reconstruct the initial information. ADVANTAGE
- Training phase is slow. DISADVANTAGE
- If a significant number of new subjects is added it is necessary to retrain the system. DISADVANTAGE
- The system is highly sensible to illumination and pose variations, to occlusions. DISADVANTAGE

NEURAL NETWORKS

The neural network aims at simulating the way brain neurons work. The input face is linearized and passed to a first network for dimensionality reduction and then to the classification network.
Each neuron is represented by a mathematical function based on probabilities. In order to recognize faces, an optimal choice might be to use a neuron for each pixel. It is clear that this approach uses too many neurons.
The solution is to use a neural network to « summarize » the image in a smaller vector. A second network performs the actual recognition.



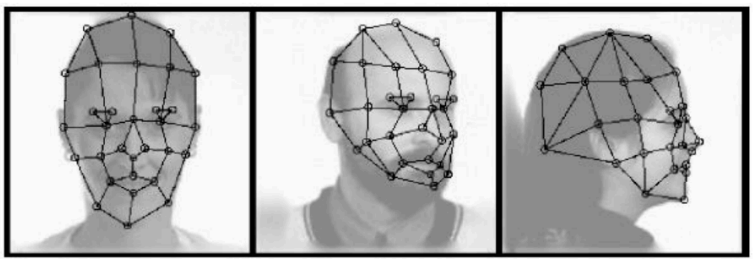
Advantages:

- This approach reduces the ambiguity among subject belonging to similar classes.
- With suitable tricks they are robust to occlusion.

Disadvantages:

- They require more than one image for training.
- Some networks are subject to a number of problems: **Overfitting** (the network has the same dimension of the input); **Overtraining** (the network loses the ability to generalize); **Database size** (when the number of subjects increases, they become inefficient).

SYSTEM BASED ON GRAPHS



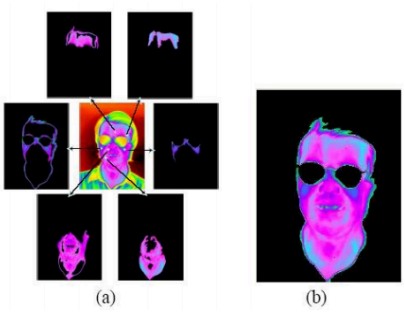
These systems use filters and localization functions to localize a set of reference points on the face. These points are connected by weighted arches so to create a graph.
Each face is associated with a graph: matching two faces means matching two graphs.

Advantages:
- They are robust with respect to pose variations.
- They are robust with respect to illumination variations.
- They do not require retraining the system.

Disadvantages:
- Training is slow.
- Testing is very slow because it requires graph matching (NP-Hard).

THERMOGRAM

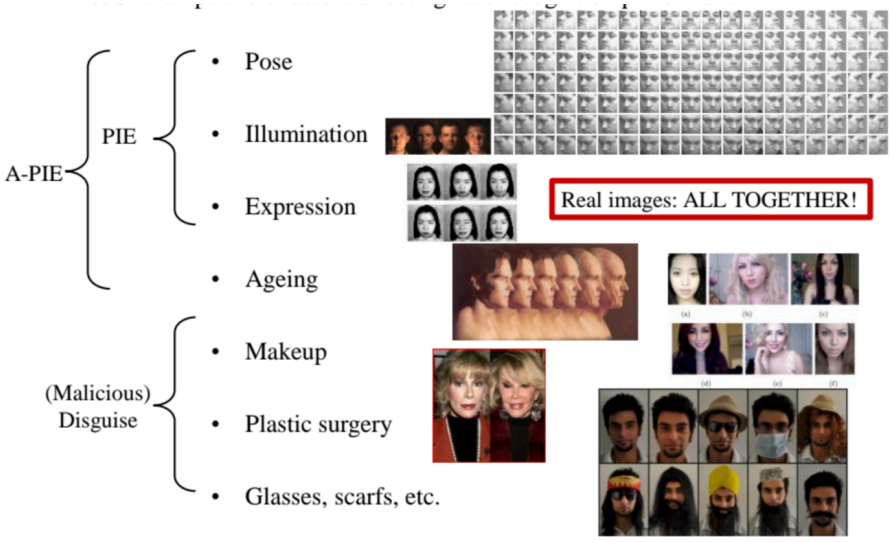
The face image is acquired through a thermal sensor. The sensor detects temperature variations from face skin. The image is segmented and indexed.



Advantages:
- They are robust with respect to illumination variations.
- They are robust with respect to time variations.
- They are efficient both indoor and outdoor.

Disadvantages:
- They require expensive capture devices.
- Capture devices are too sensitive to subject movements and provide a low resolution.
- They are affected by emotional state of the subject.
- A glass between the subject and the capture device makes the capture operation quite ineffective.

Face recognition: problems



We must know which are possible problems that we can find in general in face recognition in order to better understand which among such problems can be easily overcome using 3D techniques and which ones need a special strategy too address them even if we use 3D techniques.
The most traditional problems to be taken in the face recognition are those related to **PIE** (Pose, Illumination and Expression). Also Aging has been taken into account in more recent times (**A-PIE**).
In more recent time, also some kinds of **Disguises** have been taken into account; they are related to real world situations. We may have either **Malicious Disguised** or even **Involuntary Disguised** that can be cause by Makeup (it can completely change the appearance of a person), Plastic Surgery (if we have the face enrolled before the intervention, it could be a little bit more difficult to recognize the same person after the plastic surgery), glasses, scarfs, etc... (they are kinds of occlusion).
Which are the variations that still affect the face recognition in 3D? And which ones can be considered as solved with this kind of approach?
For example, regarding the **Pose variations**, while 2D recognition can be affected by rotation, pitch and yaw variations in one pose, this is not true with 3D.
When we carry out a recognition strategy based on 3D we actually don't use a 2D/flat image but we use a 3D model and so we can test with different poses of this 3D model because it can be rotated and its pose can be also changed according to pitch and yaw so that we can try to have the 3D model assume the same pose represented in the 2D image. So this kind of strategy is free from the dependence of subject pose.
For a similar reason also **Illumination problems** can be considered as solved, because as in the same way we can rotate the model, we can also synthetize different illuminations.
While **Expression** still affects face recognition in 3D.

Variation	2D	3D
Pose	Affected 🔴	NOT Affected 🟢
Illumination	Affected 🔴	NOT Affected 🟢
Expression	Affected 🔴	Affected 🔴
Ageing	Affected 🔴	Affected 🔴
Makeup	Affected 🔴	NOT Affected 🟢
Plastic surgery	Affected 🔴	NOT Affected 🟢
Glasses, scarfes, etc.	Affected 🔴	Affected 🔴

Why is 3D recognition still affected by Expression?
If we think especially on exaggerate expressions that are the ones more difficult to capture even in 2D, they still create problems in 3D because they can cause a real modification with the respect to a neutral 3D model. For this reason it's possible to exploit the so called **morphable model**, that can be somehow distorted in order to reproduce any kind of expression but the computational complexity increases in a dramatic way. So that may of such techniques cannot be used in real world applications.
For the same reason, also **Aging** can affect 3D strategy. According to an increasing age, it's possible that some face issues relax. So that just because the volume of a certain anatomic part can increase with age, this affects 3D strategy too.
Makeup doesn't affect 3D model at all because only consider the geometric relationships, the 3D volume determined by the face, this don't change with super-face makeup.
Plastic Surgery can affect 3D models according to the kind of weight and the extension of the intervention.
Occlusion affects 3D as it affects 2D, because when we build the model for the probe and it wears a scarf or glasses or whatever, this model is poorly matched against the the enrolled one.
We can summarize the **pros** and **cons** of **3D strategies** by saying that:

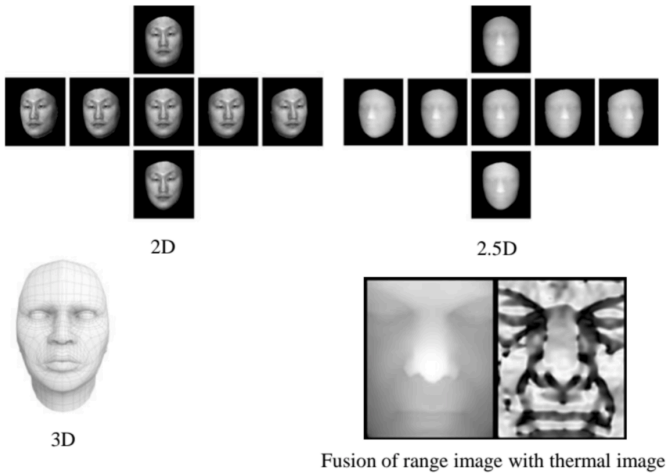
- In 3D we have much more information, the built models are much more robust to a number of distortions, there is the possibility to synthetize (approximate) 2D images from virtual 3D poses and expressions computed from a 3D model. **PROs**
- The cost of devices, computational cost of procedures, possible risk that some acquisition devices can present (for example the laser scanner is dangerous for the eye). **CONs**
- In the **middle** we have techniques that are able to approximate a 3D model from 2D image(s). So we have a kind of reverse adaptation.

Which are the possible spaces where we can represent face?

- In **2D** we can use the **Intensity Images**. In this case we have a grid of sensors that work more or less according to the principles of the visual cells win the human perceptual system; the value of each pixel is given by the intensity of the illumination reflexed in that point, which depends on the kind of surface and on the kind of illumination, and which can be expressed as a grey value or in colors according to a chosen color space (e.g., RGB).
  We can also acquire face image with **other sensors**, for example thermal images or NIR (near to infrared) images.
  We can also exploit **Preprocessed Images**; they are the result of any kind of filtering or transformation that we can done extracting features from input images (for examples are wavelets or LBP processing).
- The **2.5D** is a representation that is half way between 2D and 3D. In this case we can use **Range Image**: a 2D grid where the values of each pixel represents the distance between the point and the light source; it is usually expressed in grey levels, but cal also be expressed in some color space (typically RGB), and the name derives from the fact that values represent an information in the 3D space without building a real 3D model. We cannot extract completely 3D informations from a range image, because we can only infer distances from points to the captured device in a certain position. So this is not sufficient to estimate a complete 3D model. If we acquire more range images at fixed rotation for example, it’s possible even to extract a quite reliable 3D model that

whose accuracy can be compared also with much more expensive 3D scanners. This depends also on the quality of the range device.

- In the **3D representation**, the post used representation is the **Shaded Model**: we've a structure that is made of points and polygons connected in the 3D space; each polygon represents a very small object patch (in this case a face patch), and the smaller the patched, the better the 3D reconstruction.



Very broadly speaking, we can say that:

- 2D images encode the result of the interaction between the light coming from a source and the objects in the scene, according to their reflectance properties.
- 2.5D images and 3D models encode the result of the interaction between the (structured) light or other beam coming from a source and the objects in the scene, according to their shape properties.

The first 3D acquisition devices are the Stereoscopic cameras. The object (face) is captured by one or more stereoscopic cameras with different points of view; a set of relevant features is located on on image and the corresponding sets of features are searched for in the other images; the position of each relevant point on the depth axis is given by the geometrical relation among homologous points (identify the position of each relevant landmark across depth axis and then we can find the relationship among homologous points that help us recreating the 3D model).
We have a low cost apparatus with a medium accuracy, but this kind of devices suffer for low robustness to illumination. It's difficult to reproduce the model in real time.
Another acquisition device is the **Structured light scanner**. We have a source of light beams that have to hit the object (a single light pattern is projected along the surface); we also have a screen with a pattern that is created by a projector; when the pattern hits an object along the way, we have a deformation of the image that the camera captures back (the pattern is deformed by the 3D structure of the face and the deformation gives the measure of the depth for each point). The deformation provides also the amount of depth for each point. The capture must be repeated from different points of view in order to obtain a full 3D model.
In this case we've medium-high cost, medium-high accuracy (it depends on the accuracy of the camera and also on the kind of pattern that the projector is able to project), medium-high robustness to illumination, it's not dangerous for the eyes.
If we want a much accurate result, we have to use **Laser scanner**. The principle is more or less the same to the previous one. We have an oscillating mirror, a laser that projects its light over this mirror and the photocell that captures the reflected light. In this case we've a single laser beam that is projected along the surface (face). The beam is deformed by the 3D structure of the face and the deformation gives the measure of the

depth for each point. The capture must be repeated from different points of view to obtain a full 3D model.
In this case we have medium-high cost, high accuracy, high robustness to illumination, but it's dangerous for the eyes, so it's used to build 3D model of objects.

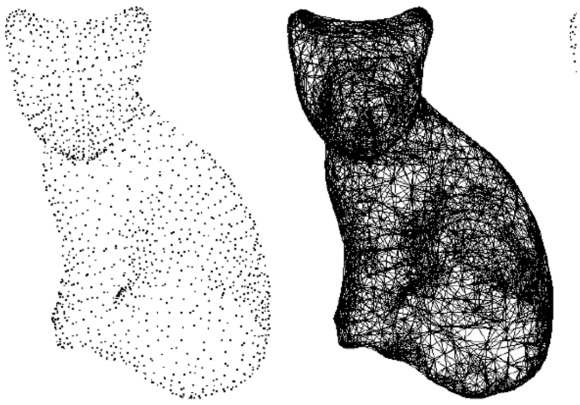
What happens when we get a 3D scan?

Due to noise, powder in the air or some kind of pollution in the air can cause occasional error in the acquisition, so that both 3D scan and 2.5D images may present both the problem of noise.
Noise can create **spikes**; looking at the pattern of distances we can either exploit a manual intervention or we can also exploit the fact that in a real surface even though we have some points that can have a larger distance as the nose with the respect to the others, by putting together the information in different poses we can find that there is not a sudden change but that the change is always gradual.
So we can apply filters or we can also try some manual interventions.
The other problem is somehow the opposite: the **presence of holes**. For example there are some case where some points are not captured and so we have 2.5D and 3D images with holes. In this case we can use the Gaussian smoothing, linear interpolation, symmetrical interpolation, morphological operators.
We can also have the problem of **smoothing**, in the sense that, even without having a spike, we can have some inaccuracy in the capture so that it may be necessary to smooth the surface.
And then also the **alignment**, because whenever we have different images, before building the 3D model the main landmarks must be perfectly aligned, otherwise we introduce distortions in the 3D model.

FROM 2.5D REPRESENTATION TO 3D MODEL

A 3D model is constructed by integrating several 2.5D face scans which are captured from different views:

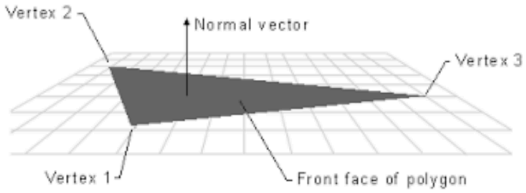
1. for each 2.5D image a cloud of 3D points is generated, with x and y coordinates equally spaced and z (depth) coordinate derived from the value in the 2.5D Image
2. the cloud of points is triangularized (a mesh of adjacent triangles is derived)



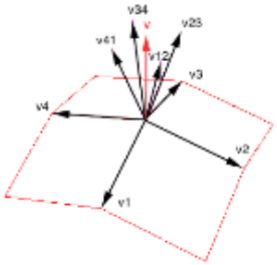
3D MODEL

A **polygon** is a sequence of coplanar points (**coplanar** means that they are on the same plan so that along this plan the normal orientation is always the same; this is not true on a real surface) connected by segments. **Coplanarity** is what causes the approximation when we represent a surface through a **mesh of polygons**: the smaller the polygons, the better the approximation. A side is a segment connecting two vertices. A polygon is composed of a set of sides (the first vertex of the first side corresponds to the second vertex of the last side). Each vertex in the mesh is adjacent to two sides at least. Each side belongs to one polygon at least.

How can we determine the orientation of a surface in the 3D space? We use the **normals** (to vertices or to polygons).



- **The normal to a polygon** is a vector perpendicular to the plane where the polygon lies: it identifies the orientation of the polygon in the space and is computed by the cross product of two vectors in the plane.
- **The normal to a vertex** is the (normalized) sum of the (unit length) normals to its adjacent polygons: it identifies the orientation of the vertex in the space



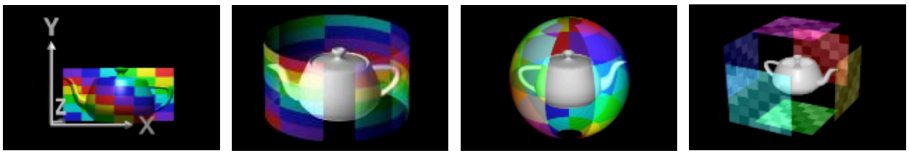
How can we compute in practice the normal vector? It's just the cross product of any two vectors in the plan. We take any two vectors; we compute the cross product and we find out the orientation of the normal vector.

When we connect points in the triangulation we want to minimize the changing of the normal passing from one point to the other. We would like to have a kind of constant normal orientation along the same triangle.

After building the **polygonal mesh** (geometrical structure) it is necessary to assign colors to vertex and/or polygons (perceptive structure).

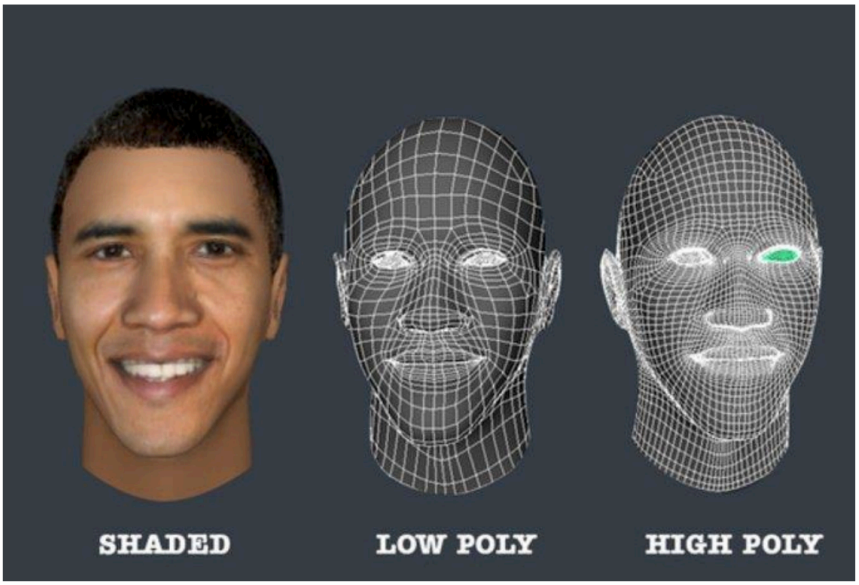
It is possible to **assign a color to a vertex** (such colors can either try to reproduce a real object or code some special information that we may extract from the scene); we can assign a color to to a polygon according to the colors of its vertices: a **uniform colo**r derived from the sum of the colors of the vertices; a **varying color** depending from the colors of the vertices and from the distances of each point of the polygon from the different vertices.

It's possible to use different kinds of projections in order to map any kind of texture over the obtained 3D model. The **Texture mapping** is used to compute position and orientation of a texture on a surface.



planar cylindrical spherical box

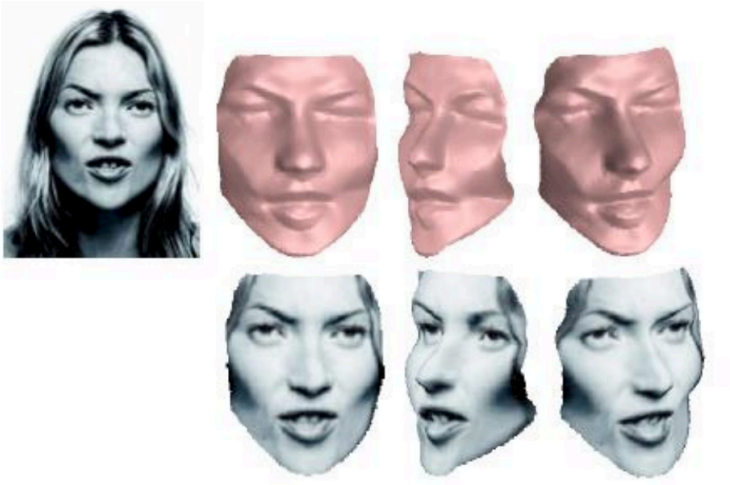
The result is different according to the kind of Texture mapping that we use. When we have a texture to map, the first **problem** to solve is: how to map each texture pixel on the correct surface point?



Higher is the number of the polygons, the better the smoothness of the surface.

FROM 2D REPRESENTATION TO 3D MODEL: SHAPE FROM SHADING

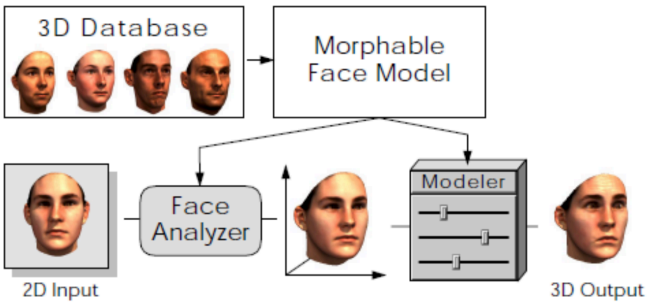
We have another way to obtain a 3D model from a scene by using the so called shape from shading. Whatever is the object, but in particular with faces, shading can be used as a cue for shape reconstruction.
We can exploit the **relationship** existing **between intensity and shape**. In practice, the computation of the map takes into account the assumption that we have a **Lambertian surface**.
It's not possible in general to recover shape from a single image, because this is not possible with pure 3D acquisition devices neither.
So we usually need more images taken in 2D and we can use statistical statistical techniques such as PCA to derive a dimensionally reduced representation for shapes in the same class.
A recent approach from **Kemelmacher-Shlizerman and Basri** in 2011 avoids representing input faces as combinations (of hundreds) of stored 3D models, and uses only the input image as a guide to ”mold” a single reference model to reach a reconstruction of the sought 3D shape.



So they use a single input image to somehow mold or morph a single reference model in order to reach a kind of reconstruction of the 3D shape.
This technique uses morphing (typical of image graphics). The procedure starts from a generic initial 3D model (**morphable model**) and arrives to a 3D model for a specific subject.
The model is called morphable model because we can change some relationship among polygons in the model in order to create different appearances of the same face or also to create different faces.

Shape and texture of the generic model are manipulated to adapt to the captured images. Morphable models also allow to synthetize face expressions approximating the possible expressions of a specific subject.
The morphable model is based on a data set of 3D faces. Morphing between faces requires full correspondence (alignment) between all of the faces.
The geometry of a face is represented by a shape vector composed of the set of X, Y, Z coordinated of the n 3D vertices $S=(X_1, Y_1, Z_1, ..., X_n, Y_n, Z_n)^T \in R^{3n}$.
A similar vector represent the texture of the face through RGB values. A morphable face model is constructed using a data set of m exemplar faces, each represented by its S_i and T_i vectors.
New shapes and textures can be obtained through a linear combination of the shapes and textures of the m exemplars:

$$\mathbf{S}_{mod} = \sum_{i=1}^{m} a_i \mathbf{S}_i \,, \quad \mathbf{T}_{mod} = \sum_{i=1}^{m} b_i \mathbf{T}_i \,, \quad \sum_{i=1}^{m} a_i = \sum_{i=1}^{m} b_i = 1.$$



We start from the 3D Database with the exemplars: we extract a Morphable Face model; when we have a 2D input, we pass it to a Face Analyzer; then there is a procedure in order to morph the Face Model extracted from the exemplars inn order to both adapt to the 2D input and to how the model can be modify the morph in order to obtain the same face in a different pose and expression.

3D FEATURES

The most important informations are normals and local and global curvatures of the face model because they express the changes in the orientation of different patches of face. 3D face recognition algorithms often work on local and global curvature of the face model.
In particular, it is possible to extract the information concerning the shape of a 3D face by analyzing the local curvature of the surface.
Examples of informations that can be extracted are:

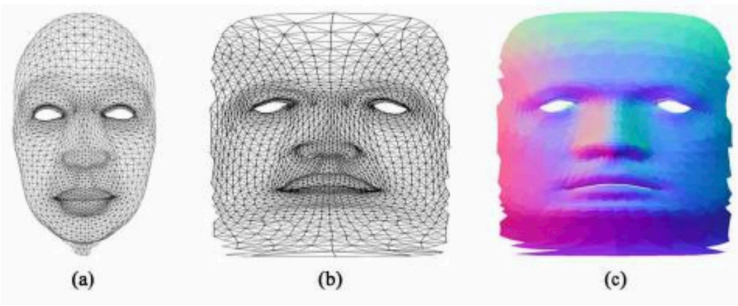
- **Crest Lines**: selecting areas with the greatest curvature; these are the lines where we can find a sharper variation in the orientation.
- **Local Curvature**: representing the local curvature with a color; in Local Curvature such curvature can be expressed with different colors so that lighter the color, the highest the degree of curvature.
- **Local Features**: segmenting the face into regions of interest; we can segment the face into regions of interest and extract features vector from the single region, for example from the eyes, from the nose, etc… This means that we also annotate the 3D model with. relevant landmarks that can be useful in order to identify the patches.

We have to find out, before being able to reliably extract features, how to carry out a kind of landmarks alignment because otherwise the information about local curvatures and normal orientations can refer to different parts of face so that even a slight difference can hinder a good accuracy of the result.

Points can be either located on the 2D image and then projected over the 3D model or also we can process the 3D model directly in order to extract the landmarks.
The alignment can be carried out in more steps, in the sense that we have a quite precise procedure that is **Iterative Closest Point** (**ICP**) **algorithm**. This is widely used for geometric alignment of three dimensional models when an initial estimate of the relative pose is known.
We must start from an initial estimate because the better the initial estimate, the lower the computational burden created by ICP (because ICP is extremely expensive from a computational point of view). So the **problem** moves to find out a good initial estimate in order to decrease the complexity of the ICP algorithm in itself.
We can find an initial match between the surfaces by matching curves, relevant points, points of maximum changing, orientation and so on. Then we can compute the distance obtained by summing up the distances between homologous point in the two surfaces that have been roughly aligned in this way. Then we compute the transformation which minimizes such distance.
Perform the transformation and reiterate the procedure until the distance is less than a threshold.

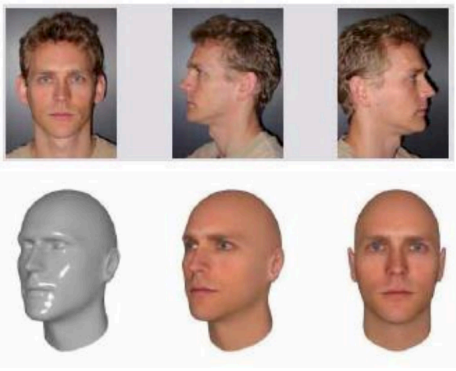
3D FACE RECOGNITION WITH NORMAL MAPS

We start from the acquisition and the generation of the face.
Then we have to project geometry from 3D space to 2D space according to basic projection rules.
We can generate the so called **Normal Map**. This Normal Map can be stored as a regular RGB image where the RGB components correspond to the X, Y, and Z coordinates, respectively, of the surface normal.
This is a way to carry out a kind of inverse mapping from the 3D model to a 2D image that can be compared with similar images as well.



We can use these kind of images for comparison when we want to compare two 3D model.
We can exploit normal maps for 3D face recognition so once we have the RGB mapping of the components that is represented by the intensity of the color, then we can carry out a study of the trend followed by normal maps and we can obtain a kind of compact information regarding the 3D surface.
The **components of the normal** (unit vectors) are sampled with a triple RGB. The length of the vector component is represented by the intensity of the color.
Using the normal map we obtain a two-dimensional representation of three-dimensional information. The information about the curvature of a model is represented by the set of the surface normals. Reading and processing a 2D image is much faster than reading and processing a 3D model.
The **difference map** is the difference image between two normal maps. Each pixel of the difference map represents the angular distance between two normal maps in that point.

3D FACE RECOGNITION VIA MORPHABLE MODELS: FACEGEN MODELLER



We can use morphable models. There is a popular generator that is **FaceGen** that is able to shape faces from a number of 2D images.
FaceGen Modeller is a commercial tool able to generate the 3D model of the face of an individual “instrumented” with an underlying structure capable of simulating the "dynamics" of the face.
The algorithm is based on fitting a model "generic" (**morphable-model**) to the shape and color of the final model of the subject to be captured.
The adaptation of the morphable model to the final model is driven by a set of facial features extracted directly from the photos.
Morphable-models are a powerful tool, allowing you to generate synthetic facial expressions.
Synthetic facial expressions can be used to obtain a good approximation or real facial expressions that an individual may have during the process of acquisition, and can be used in some cases to add samples to the gallery or to reproduce probe expression on the fly but also to simulate aging.

3D FACE RECOGNITION USING ISO-GEODESIC STRIPES

Once we have distances, we can identify stripes with the same **geodesic distance**. In presence of an expression change, we can point positions in the 3D surface can change. We can observe that geodesic distances (we can assume that the geodesic is the shortest path between two points in a curved space) are slightly **affected by changes of facial expressions**.The geometry of convex face regions is less affected by changes of facial expressions than concave regions.
3D face recognition is computationally expensive.
Some studies have demonstrated that the Euclidean and geodesic distances and the angles between as many as 47 fiducial points may suffice to capture most of relevant facial information.
However in practice automatic detection of fiducial points is difficult and the full 3D face surface information must be exploited for matching.
The conversion of 3D scans to efficient and meaningful descriptors of face structure is crucial to perform fast processing and particularly to permit indexing over large datasets for identification.
In this approach, all the points of the face are taken into account so that the complete geometrical information of the 3D face model is exploited.
The relevant information is encoded into a compact representation in the form of a graph and face recognition is finally reduced to matching the graphs.
Face graphs have a fixed number of nodes, that respectively represent **iso-geodesic facial stripes** of equal width and increasing distance from the nose tip.
Arcs between pairs of nodes are annotated with descriptors referred to as **3D Weighted Walkthroughs** (**3DWWs**), that capture the mutual spatial displacement between all the pairs of points of the corresponding stripes and show smooth changes of their values as the positions of face points change.
This representation has the great **advantage** of a very efficient computation of face descriptors and a very efficient matching operation for face recognition.

WEIGHTED WALKTHROUGHS IN 2D AND 3D

In a two-dimensional Cartesian reference system with X, Y coordinate axes, given two generic points a = (x_a; y_a) and b = (x_b; y_b), their projections on each of the two axes can take three different orders (b with respect to a): before, coincident, or after, for a total number of nine possible cases of two-dimensional displacements between a and b.
Each displacement can be encoded by a pair of indexes <i, j>, with i and j taking values in {-1, 0, +1}:

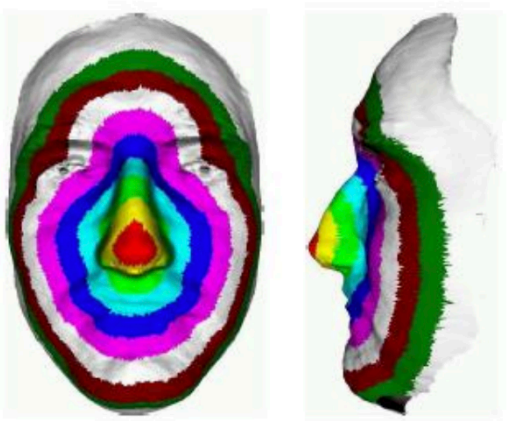
$$i = \begin{cases} -1 & x_b < x_a \\ 0 & x_b = x_a \\ +1 & x_b > x_a \end{cases} \qquad j = \begin{cases} -1 & y_b < y_a \\ 0 & y_b = y_a \\ +1 & y_b > y_a, \end{cases}$$

A **displacement** can be regarded as a walkthrough from a to b. Given two continuous regions A and B, points of A can be connected to points of B by walkthroughs.
The number of unique pairs (a, b) with a $\in$ A and b $\in$ B that are connected by the same walkthrough <i, j> can be measured and is denoted as $w_{i,j}$ (A,B).
The 3×3 matrix w(A,B) of the weights is referred to as **2D Weighted Walkthrough** (**2DWW**), and can be used to model the relative displacement between the two sets.
Intuitively, 2D Weighted Walkthroughs can measure the mutual spatial distributions of the masses of two 2D regions.
Extension to 3D of the above takes to the definition of 3DWW.
Directional indexes are aggregate measures that take values in [0; 1] and account for directional information between two 3D spatial entities.
Directional Indexes can be directly derived from the weights of the 3DWW matrix.

ISO-GEODESIC STRIPES

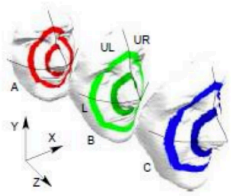
Each face is partitioned into a fixed number of iso-geodesic stripes of equal width, pseudo-concentric and centered on the nose tip fiducial point. The mutual displacement between each pair of stripes is measured by computing the 3DWWs between all the pairs of points of the two stripes.
Face stripes are obtained by:

- computing the normalized geodesic distance γ between each face point and the nose tip (using the Dijkstra's algorithm applied to the points of the surface).
- quantizing γ values into N intervals c_1, ..., c_N, so that the i-th stripe collects all the face points that have values of γ within the interval c_i
- the normalization factor is the Euclidean eyes-to-nose distance (i.e., sum of distances between the nose tip and the two endocanthions): this guarantees invariance with respect to scaling and expression changes

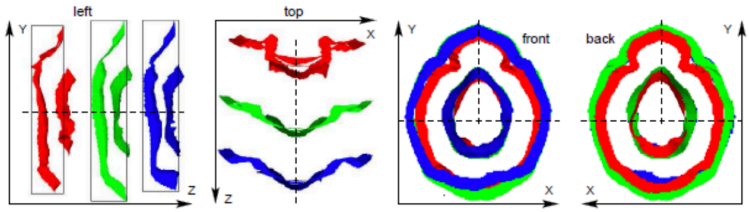


3DWW FEATURES

Under appropriate design choices, face partitioning into iso-geodesic stripes of equal width and calculation of 3DWWs between stripes provides an effective representation of 3D faces allowing to distinguish facial differences due to different facial traits of different individuals from differences induced by facial expressions of the same individual.



(a)



(b)

Index	A neutral	B neutral	C smile
w_H	0.45	0.49	0.49
w_V	0.82	0.77	0.77
w_D	0.02	0.12	0.13

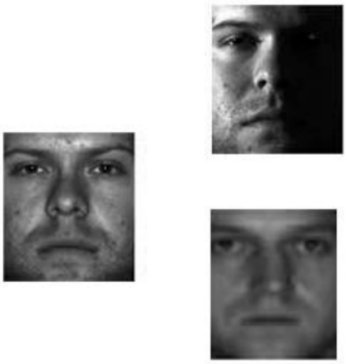
Geodesic distances between two facial points keep sufficiently stable under expression changes; the large majority of the points of each stripe still remain within the same stripe, even under facial expression changes.
Only few parts of the face are affected by deformations due to expressions.

FACE RECOGNITION - EVALUATION

Which are the open problems in Face Recognition?
First of all PIE (Pose, Illumination, Expression); also Occlusion, Time and Age can deeply modify a face appearance; Make up and accessories may hinder recognition; recognition after plastic surgery or after gender change or after getting or loosing weight.

ILLUMINATION

Illumination conditions may significantly change during the day, across different days, indoor or outdoor. Direct lights on the face produce shadows and hyper illuminated zones.



Due to such variations the feature vector associated to a subject in different illumination conditions may result closer to that of a different subject with a similar illumination, than to that of the same subject with a different illumination.
We may address this problem by choosing a reasonable acceptance threshold; but in the case of identification open set, whatever is the threshold, there is also a ranking and so the rank of the image belonging to the wrong person may be higher than the rank of the image belonging to the right person if there are present shadows.
There are three kinds of **algorithms** aims at addressing illumination variations in FR:

- **Shape from Shading**: starting from the grey levels in one or more images they extract the 3D face shape and also to project it into 2D in order to get a better representation of the candidate face to be matched with the gallery
- **Representation Based Methods**: they use classifiers that by their intrinsic nature are robust with respect to illumination variations. LBP cannot be considered among those representation based methods because in the region of the shadow, the values returned by LBP will be completely unreliable.
- **Generative Methods**: starting from a 3D face model they generate a wide set of images with the highest possible number of illumination variations, that are used afterwards for the enrolling of the subject.

POSE

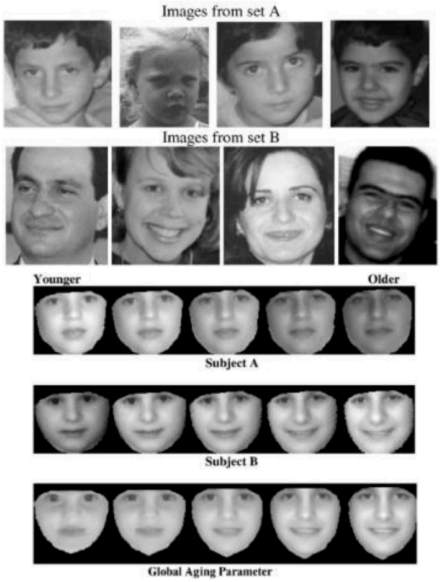
The most simple and intuitive approach is to adopt a multiview system.
Multiview system is a special class of multiple instance system; we may enroll the subject under different poses and each template extracted in this way enters in the gallery as a different instance to be matched against.
On the other hand we've also **Pose correction systems**. Pose correction systems can be based on 3D modeling so that once we've a 3D model, we can rotate and modify and morph it in order to get a template to compare which is more similar to the incoming probe; we may also try to create a kind of canonical form.

OCCLUSIONS

Many types of occlusions can partially hide the face: hair, sunglasses; scarfs; make-up; etc…
So we have many kinds of occlusions that may affect the face appearance in different ways.
Recognition algorithms must be robust to such occlusions. When we have a large occlusion, there is no way to reliably fill the gap because in any case we're reproducing a non-existent information and whatever is the strategy that we use, we may introduce a source of error.
Most approaches rely on **localized processing**, so that local occlusions do not affect the classifier in other regions. In practice, in these cases, instead of trying to correct, the most popular approaches deal with first **detection of the occlusion**: once the occlusion has been detected, especially exploiting patch wise comparison (divide the face into patches and only exploit the available ones), then we can use the occlusion like a "don't care" region and try to carry out the comparison only using the rest of the face image.

TIME AND AGE VARIATIONS

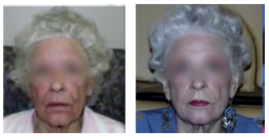
Time variations (even only one week) between two images of a same subject, may reduce the performance of a recognition system.
Thermogram is more robust to such variations.
Age variations are even more **critical**.
A system robust to age variations is useful in applications like identification of wanted or missing persons. In this case we try to mimic the aging process. The basis are ethnographic studies and the study of how some relevant somatic regions and anatomical regions of face develop.



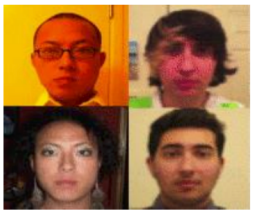
MORE VARIATIONS

- Make up

- Plastic Surgery



- Gender Change



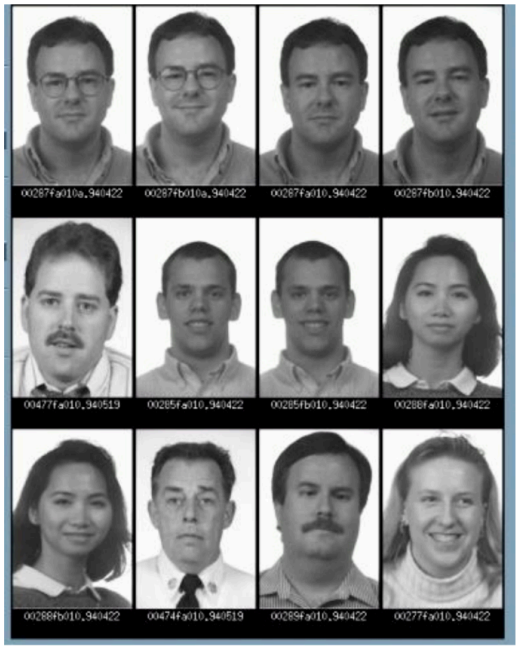
- Weight Change

SYSTEM COMPARISON

It is not only necessary to be able to evaluate the accuracy of the performance of a certain system, but it is also important to be able to compare the performance of different systems.
It can be done according to the kind of applications that we have, because we can have for example different measures when verification is entailed or when identification is entailed.
We must take into account that the measures like FAR, FRR, CMS, ... are not sufficient for a complete evaluation of an algorithm since they are too much untied from the real operation context.
We must also consider:

- Number of used datasets: it's necessary in order to evaluate the generalizability of the methods that we've proposed.
- Image size: large and high quality images provide more information; but higher is the resolution, the higher is the noise that can affect the sensor.
- Probe and Gallery size: we must work with a reasonable size in order to guarantee the generalizability of our methods.
- Number and level of tolerated variations: they depend on the kind of application that we're designing. Because if we can assume for example that the person will always present a face in normal pose with a neutral expression and a good illumination, maybe we can get rid of higher computational demanding techniques to remain with something simpler.

Specific protocols have been and continue to be devised. New datasets are created to serve as benchmarks.



FERET PROTOCOL

Before the FERET database was created, a large number of papers reported outstanding recognition results (usually >95% correct recognition) on limited-size databases (usually <50 individuals). Only a few of these algorithms used a common database, let alone met the desirable goal of being evaluated on a standard testing protocol that included separate training and testing sets. As a consequence, there was no method to make informed comparisons among various algorithms. The FERET database made it possible for researchers to develop algorithms on a common database and to report results in the literature using this database.
The independently administered FERET evaluations allowed for a direct quantitative assessment of the relative strengths and weaknesses of different approaches.
The **first FERET** evaluation took place in August 1994 (**Aug94 evaluation**) and it was designed to measure performance on algorithms that could automatically locate, normalize, and identify faces from a database: the first subtest examined the ability of algorithms to recognize faces from a gallery of 316 individuals; the second subtest was the false-alarm test, which measured how well an algorithm rejects faces not in the gallery; the third subtest baselined the effects of pose changes on performance.

The **second FERET evaluation** took place in March 1995 (**Mar95 evaluation**). The goal was to measure progress since the initial FERET evaluation, and to evaluate these algorithms on larger galleries (817 individuals). In this evaluation probe sets contained **duplicate images**: a duplicate image was defined as an image of a person whose corresponding gallery image was taken on a different date.
The **third**, and final, **FERET evaluations** took place in September 1996 (**Sep96 evaluation**). The new evaluation protocol required algorithms to match a set of 3323 images against a set of 3816 images (algorithms had to perform approximately 12.6 million matches). The new protocol design allowed the determination of performance scores for multiple galleries and probe sets, and perform a more detailed performance analysis.
Results were reported for different cases:

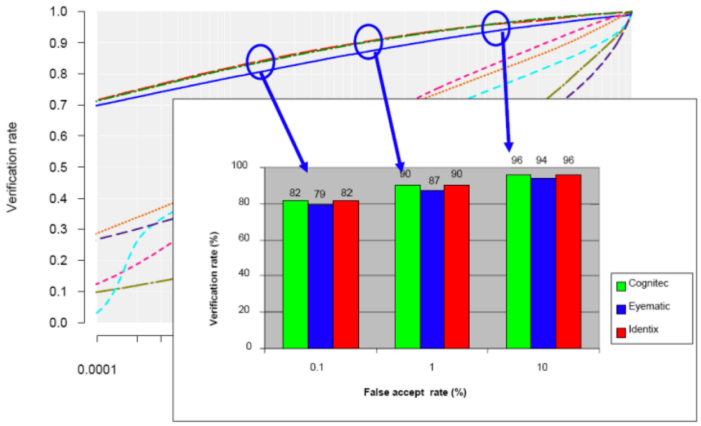
1. the gallery and probe images of a person were taken on the same day under the same lighting conditions
2. the gallery and probe images of a person were taken on different days
3. the gallery and probe images of a person were taken over a year apart
4. the gallery and probe images of a person were taken on the same day, but with different lighting conditions

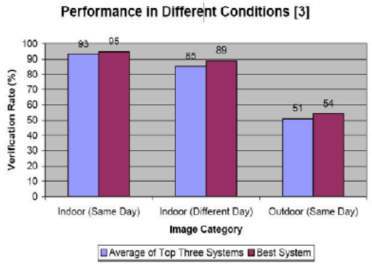
There were **two versions of the September 1996 evaluation**:

- The **first tested** partially automatic algorithms by providing the images with the coordinates of the center of the eyes.
- The **second tested** fully automatic algorithms by providing the images only.

FACE RECOGNITION VENDOR TEST (FRVT)

After FERET evaluations, the **Face Recognition Vendor Test** (**FRVT**) series has been organized.
The **FRVT 2006** measured performance with **sequestered data** (data not previously seen by the researchers or developers).
A standard dataset and test methodology was employed so that all participants were evenly evaluated.
The government provided both the test data and the test environment to participants. The test environment was called the **Biometric Experimentation Environment** (**BEE**). The BEE was the FRVT 2006 infrastructure. It allowed the experimenter to focus on the experiment by simplifying test data management, experiment configuration, and the processing of results. Present challenges follow the BEE philosophy.





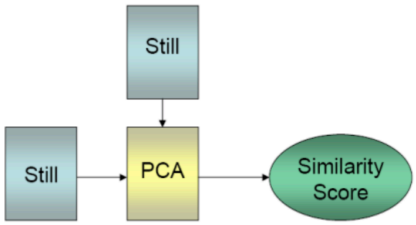
FRVT 2002 demonstrated that the performance of the 2D algorithms are drastically reduced when using images of the same subject yet taken with very different illuminations (indoor & outdoor images even if acquired on the same day).

FRVT 2000 and **FRVT 2002** demonstrated that another difficult task for 2D facial recognition systems is to recognize faces that are not represented in frontal pose.
Most systems provide good 2D facial recognition performance when the image is frontal. If the pose of the face changes (both horizontally and/or vertically) the performance decrease.

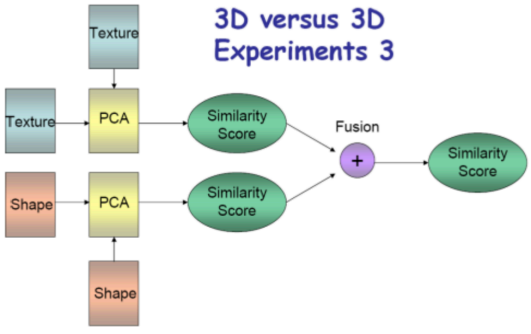
Face Recognition Grand Challenge (FRGC)

The **Face Recognition Grand Challenge** (**FRGC**) is designed to achieve an increase in performance of 2D and 3D recognition algorithms.
It is a "**challenge**" between the algorithms presented by different groups of researchers.
The organizers have provided the researchers with a dataset with over 50,000 records divided into two partitions (Training Set and Validation Set) and a set of testing procedures in order to make the results of the algorithms of facial recognition comparable.
Once fixed FAR at 0.1% the existing systems were able to achieve a verification rate of 80%. FRGC aimed at reliably achieving a verification rate of 98%. This required a high number of images and tests:

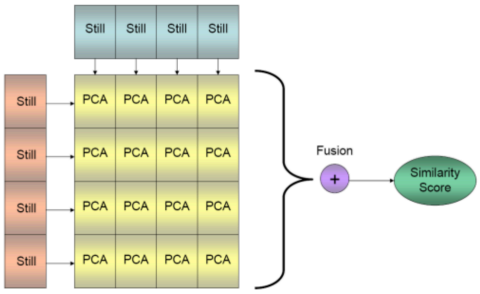
FRGC also included 3D models. There was also a number of testing sessions that aimed at comparing the performance of 3D versus 2D applications.
In Controlled indoor still versus indoor still, PCA was used as baseline (PCA is the lowest level of accuracy that we can achieve). The image (test) of a subject is matched against with the images (gallery) of all subjects in the database, and a similarity measure is computed. This is repeated for each test subject, to obtain a similarity matrix.



In the experiment where indoor multi-still versus indoor multi-still, the texture and shape of two 3D models are matched separately, and then the single similarities are fused into a single value. The 3D model (test) of a subject is compared with the models (gallery) of all subjects in the database to obtain a similarity measure. This is repeated for each subject to obtain a similarity matrix.



In the case 3D vs 3D, Matching between two subjects is performed by comparing N>1 images (test) with M>1 images (gallery), and the obtained values are fused into a single one.



FACE RECOGNITION - SPOOFING

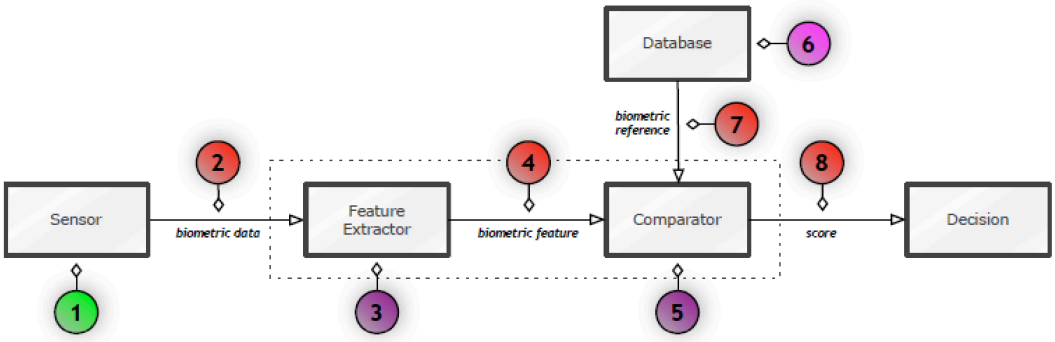
In the context of **network security**, a **spoofing attack** is a situation in which one person or program successfully masquerades as another by falsifying data (IP address, e-mail address, etc.) thereby gaining an illegitimate advantage.
It's a problem related with the ability to detect **counterfeit samples** that can be presented to a system.
Biometric spoofing is the act of fooling a biometric application, by using a copy or performing an imitation of the biometric trait that is used by that system in order to legitimately authenticate a user.
In order to successfully carry out a **presentation attack**, it is necessary to know which is the biometric trait or which are the biometric traits underline the authentication process.
Biometric spoofing attack is carried out by presenting a counterfeit biometric trait to the system to fool it pretending to be a genuine user.
In Biometric Spoofing we consider different kinds of attack, where the attacker actively acts in order to acquire the appearance of the attacked person. We have to also distinguish Biometric Spoofing from **Camouflage** or **Disguise** because in the case we've a different problem: the attack is carried out presenting an artifact biometric trait to the system to fool it pretending not to be oneself. So in that case we don't want to be recognized.

If we introduce extraneous elements on the face we may have that the detection process fails.
Even with **other biometric traits** we can observe the same difference.

Spoofing

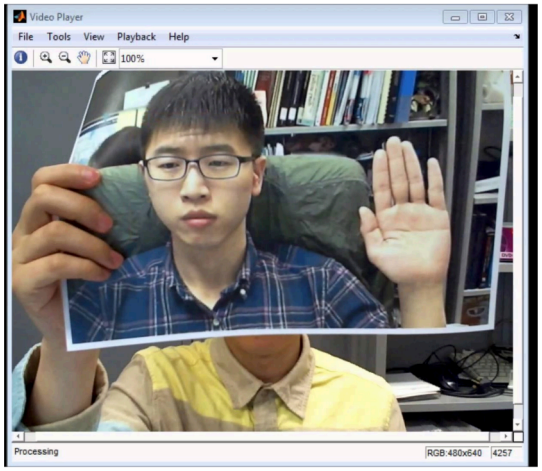
Camouflage

A biometric system can be attacked in different points: in the transmission channel; in the feature extractor; in the comparator; it's possible to inject data over of each of such channels in order to force the result that we want; it is also possible to attack the Database with templates that don't belong to enrolled people.
Especially when there is an automatic update of the gallery, in order to address the change of appearance of a person, it is possible to "poison" the sub-gallery for that subject by injecting images that are at the beginning similar to those of the real person and then, playing with the acceptance threshold, injecting different images up to be able to be recognized even with one of those.



All of these kinds of attacks along the channels, the database and so on and so forth, can be considered as **Indirect Attacks**. Defending from these kinds of attacks is a matter of cybersecurity.
Another kind of attacks are the **Direct Attacks**: the spoofing attacks also popular as presentation attacks.
In this case, our final aim is not to recognize the person, but rather to detect the attack so that we can proceed with a countermeasure.
We can do a **Classification** of spoofing attacks:

- **2D Spoofs**: the attack is carried out by using a 2D surface like a photo or like a video that is played once we have somehow recorded the attacked person. **Photo attacks** can use either hard copy, hard copy with a eye whole, screen. An alternative for 2D spoof is to exploit **Video Attacks**. In that case we recorded the person that we want to attack using different screen resolutions and then present this video instead of our true face. This is also called "**Replay Attack**".
- **3D Spoofs**: require that there is not a plain surface but a three dimensional surface and in general masks are used for the attack (**Mask attacks**).



In the general, the spoofing attacks are **dependent by the biometric trait**. In case of face, we have a variety of possible attacks.
For a human operator it is trivial to detect such kind of spoof attack; especially when the image that is shown to the system does not occupy the overall screen.
An automatic system is designed in order to identify region of interest containing face; since in this case the region of interest containing the face is present, there are all the things which automatic system cares about. Unless we extend the functionality of the system with an anti-spoofing procedure, it is quite easy to carry out this kind of attack.
For each kind of attack there are different anti-spoofing techniques that are mostly based on liveness detection. In general, what we try to do, a part from studying the reflection pattern of the surface and the microtexture, is to "challenge" the user in order to detect the kind of reaction of the face that we've in front.
We've anti-spoofing techniques that work at **Sensor-level** (**Hardware Based**); in practice they exploit the intrinsic properties of a living body including the kind of **reflectance** that is producing by a living body, **involuntary signals** of a living body (for example micr movements of the eyes) that are completely absent in a photo or in a mask.

An example of anti-spoofing technique that takes advantage of these kind of movements is **Eye Blink**. In fact, the eye blinking is a natural involuntary movement that can be detected in order to distinguish a real person from a photo.
An the other hand, we can have a **voluntary response** from the so called **"Challenge-Response" strategy**. In this case we can ask to a person that is presenting a probe to make a certain kind of expression or movement. If what we expected doesn't happen on the face that is shown to the system, we can conclude that this is a spoof attack.
Other kinds of anti-spoofing techniques are carried out at **Feature Extractor level** (**Software based**) and the can be either static or dynamic.
We study **static** for example for the microtexture of the acquired image; with **dynamic** we study the kind of dynamic features that a certain movement can produce when movement is naturally carried out.
We can have also a **Score level fusion** where we can either fuse anti-spoofing and recognition in a single module or carry out anti-spoofing in advance and proceed with the recognition only if the anti-spoofing returns a genuine response or there are other kinds of possibilities.

WHICH ARE THE MAIN APPROACHES FOR FACE?

- **Print Attack**
  This one requires the lower effort from the point of view of the attacker because it is sufficient to print an image.
  Print attack requires also the lower effort from the point of view of the countermeasure because: we can study the **texture**; some special features that can be extracted from **shape** and change whether the shape is a real 3D shape or just something printed; we can exploit the **movement** (if we ask to rotate the face for example, a printed face cannot respond to this); we can also use a **sensor fingerprint**.
- **Video**
  Try to find out the attack in a video.
- **3D Mask**
  This is the most difficult one both from the point of view of the attacker and of the defender.

2D PRINT ATTACK

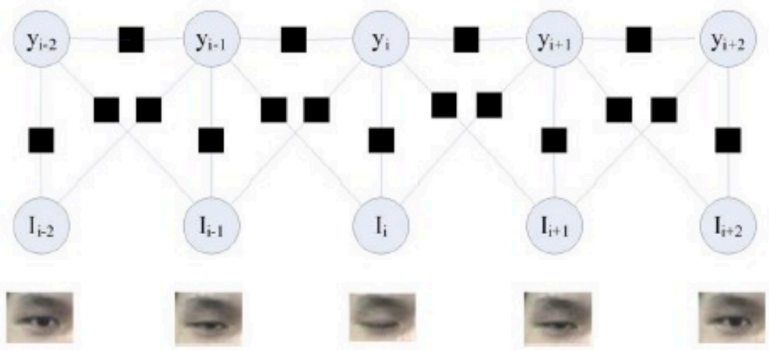
One of the most intuitive approaches to 2D Print Attach is **Liveness Detection**. In practice, the **essential difference between the live face and photograph** is that a live face is a fully three dimensional object while a photograph could be considered as a two dimensional planar structure.
This means that we can use structure from motion to derive kind of depth information to distinguish a live person from a still photo.
The **disadvantages of depth information** are that, it is hard to estimate depth information when head is still (we have to ask for a movement, otherwise it is very difficult to carry out a liveness detection in this trivial way), and the estimate is very **sensitive to noise and lighting** (if we're in a dark environment, this may help the attacker to hide some feature change with the respect to a real face).
It is possible to compute the **optical flow** (this is a technique that tries to extract the motion vector by comparing the position of each pixel in one frame and in the following one) on the input video to obtain the information of face motion for liveness judgment, but it is **vulnerable to photo motion** in depth and **photo bending**.

A possible **multimodal approach** fuses face-voice against spoofing exploiting the lip movement during speech. This kind of method needs voice recorder.
Among the earliest approaches, it is possible to consider those relying on eye blinking analysis because it relies on the natural pattern that is produced on a regular basis in eye dynamic.
Eyeblink is a physiological activity of rapid closing and opening of the eyelid.
Blink speed can vary with fatigue, emotional stress, amount of sleep, eye injury, medication, or disease, but the spontaneous resting blink rate of a human being is nearly from 15 to 30 eyeblinks per minute (a blink every 2 to 4 seconds) and the average blink lasts about 250 milliseconds. It's a movement that cannot be avoid at all.
A current generic camera can easily capture the face video with not less than 15 fps, i.e. the frame interval is not more than 70 milliseconds, therefore it can capture two or more frames for each blink when the face looks into the camera.
There are many ways to model eyeblink behaviour, for example the work by Pan et al. models eyeblink behaviors by an undirected **Conditional Random Field framework**, incorporated with a discriminative measure of eye states.
An eyeblink activity can be represented by an image sequence S consisting of T images, where $S = \{I_i, i = 1, \ldots, T\}$.
The typical eye states are **opening** and **closing**. In addition, there is an **ambiguous state** when blinking from open state to close or from close state to open. It is possible to define a **three-state set for eye**s, **Q = {α : open, γ : close, β : ambiguous}**.
A typical blink activity can be described as a state change pattern of α → β → γ → β → α.
Suppose that S is a random variable over observation sequences to be labeled, and Y is a random variable over the corresponding label sequences to be predicted, all of components y_i of Y are assumed to range over a finite label set Q. We can taking into account that some stage changes can be lost due the frame rate of the camera.
Let G = (V,E) be a graph and Y is indexed by the vertices of G. Then (Y, S) is called a **conditional random field** (**CRF**), when conditioned on S, the random variables Y and S obey the Markov property w.r.t. the graph:

$$p(y_v|S, y_w, u = v) = p(y_v|S, y_w, u \sim v)$$

where $u \sim v$ means that u and v are neighbors in G.

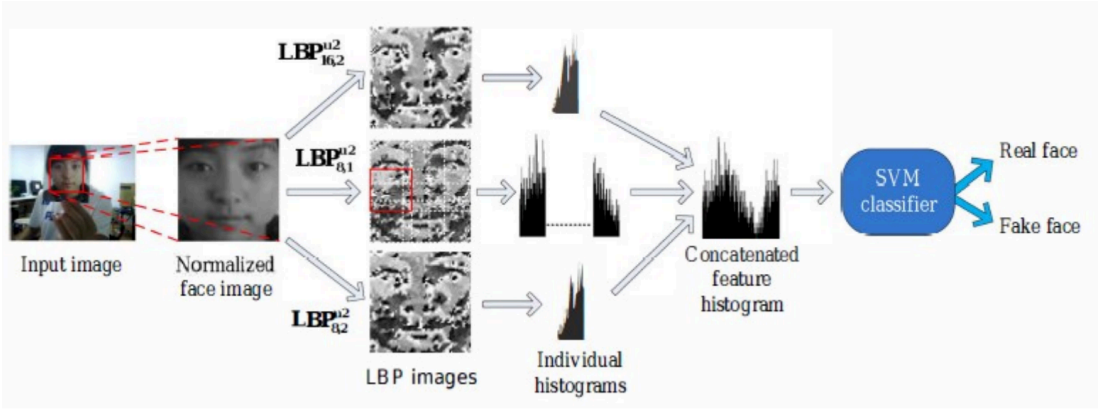
So we can predict a sequence of labels and try to compute the probability of the next state given what we have observed so far. We have conditional probabilities that are driven by what is the possibility of finding a certain state given that we have already observed a certain sequence.



All of these are used to train an Adaboost algorithm in order to be able precisely identify eye closely, in order to start the detection of the eye blinking pattern starting from close state.
An other possibility is based on **Micro-texture analysis**.
This is especially efficient when we have 2D Print attacks or Photo attacks because any kind of photographic paper or printing paper has a kind of micro-texture that can be not visible by eyes but can be detected by using texture features.
The proposal assumes that face prints usually contain printing quality defects that can be well detected using texture features.

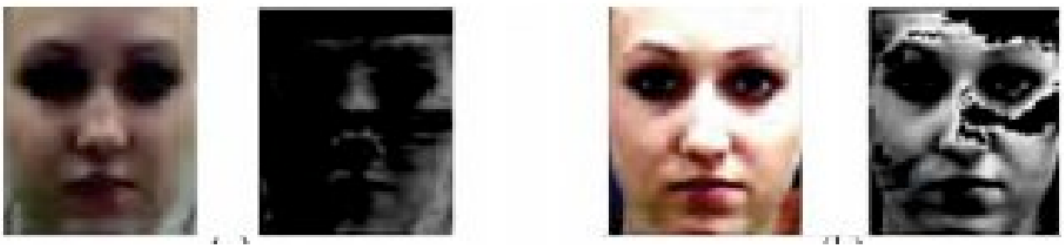
Human faces and prints reflect light in different ways because a human face is a complex non rigid 3D object (so it reflects light in different directions) whereas a photograph is a planar rigid object (different specular reflections and shades).
The **surface properties** of real faces and prints, e.g. pigments, are also different because natural pigments are different from ink pigments. These last ones contain some metallic components so this affects the way light is reflected.
The work exploits **multi-scale local binary patterns** (**LBP**). The strategy that is used in multi-scale local binary patterns is more or less the same but much more simpler to adopt than the bank of wavelets with different frequencies. So multi-scale local binary patterns are computed by using windows of different size, for example: we start from the 3x3 window with 8 neighbors, then we increase radius from 1 to 2 and this gives us a 5x5 LBP window (also in this case we can just consider 8 neighbors in the 8 principle directions; but another interesting variation is to exploit all the 16 neighbors in 5x5 window).
As a further **advantage**, the same texture features that are used for spoofing detection can also be used for face recognition.
The vectors in the feature space are then fed to an **SVM classifier** which determines whether the micro-texture patterns characterize a live person or a fake image.

In this example we've a live face and a face print in the original space and the corresponding LBP images (basic LBP).
We can notice that the printed photo looks quite similar to the image of the live face whereas the LBP images depict some differences.
In this case it's used the **Uniform LBP** (this limits the numbers of bins to the uniform LBP patterns, that are those where we can find 0 or 2 variations from 0 to 1 or vice versa).
The face is first detected, cropped and normalized into a 64 x 64 pixel image (this is a size that is sufficient to detect the relevant details of the image without a large computation efforts).
$LBP^{u2}_{8,1}$ operator (u means uniform LBP) is applied on the normalized face image and the resulting LBP face image is divided into 3 x 3 overlapping regions (with an overlapping size of 14 pixels).
The local 59-bin histograms from each region (because we're considering only the uniform LBP codes) are computed and collected into a single 531-bin histogram.
The the approach computes two other histograms from the whole face image by using **$LBP^{u2}_{8,2}$** and **$LBP^{u2}_{16,2}$** operators, yielding 59-bin and 243-bin histograms that are added to the 531-bin histogram previously computed (this time there are no partitions into regions).
At the end, the **length of the final enhanced feature histogram is 833** (i.e. 531 +59+243).
This histogram is used to train a SVM classifier with radial basis function kernel and it is used exploiting a set of **positive** (**real faces**) and **negative** (**fake faces**) samples.



Another important approach is the **Captured-Recaptured approach**. The work by Kose and Dugelay exploits a rotation invariant **LBP variance** (**LBPV** is something that is able to measure how LBP changes in a long surface) based method together with a pre-processing step of **Difference of Gaussian** (**DoG**) filtering (when we applying Difference of Gaussian filters what survives are the most contrasted regions of the image).
In practice we've to consider what does it means Captured-Recaptured. We must consider that all the distortions that are present when we capture an image affect the face image when it passes through the camera system. Such distortions are applied twice when we take the photo of a person and then when we print it. In practice, when we use a photo taken by a photo, like it may happen when using a photo taken on the internet, we have a much lower image quality compered to the capture of a real face image.
The **Difference of Gaussian filter is used as pre-processing step** to obtain a special frequency band (that is exactly what survive from this Difference of Gaussian) which gives considerable information to discriminate between real and photo images. So selecting in a feasible way the Difference of Gaussian filter that we apply, we can isolate a certain band of frequency that is able to provide a lot of informations about photo images.
A recaptured face image has less sharpness (lower image quality) when compared to captured face image; therefore recaptured image contains less high frequency components. This fact can be observed by analysing the 2D Fourier spectra of real face and photo face images.
We again compute the LBP codes and the LBP histograms and then we exploit the variation LBP (LBPV) that is used to add contrast information to the LBP histogram.
In practice, the LBPV computes the variance from a local region and accumulates it into the LBP bin as the weighting factors.
The quadratic means of histogram distances (chi-square) between a new probe and genuine and fake model sets of histograms are computed and compared to classify the new probe.
Another approach is the **Gaze Stability**. There is an algorithm proposed by Ali et al. that is based on the assumption that the spatial and temporal coordination of the movements of eye, head and (possibly) hand involved in the task of following of a visual stimulus is significantly different when a genuine attempt is made compared with certain types of spoof attempts.
In a **challenge response strategy**, when the system asks us or to the genuine subject to rotate the head, the kind of temporal coordination between the movement of the eyes, the movement of the head (if they are possibly involved) is different when if the person is looking at the screen through the holes in a mask.
The task, or the so called "challenge", requires head/eye fixations on a simple target that appears on a screen in front of the user.

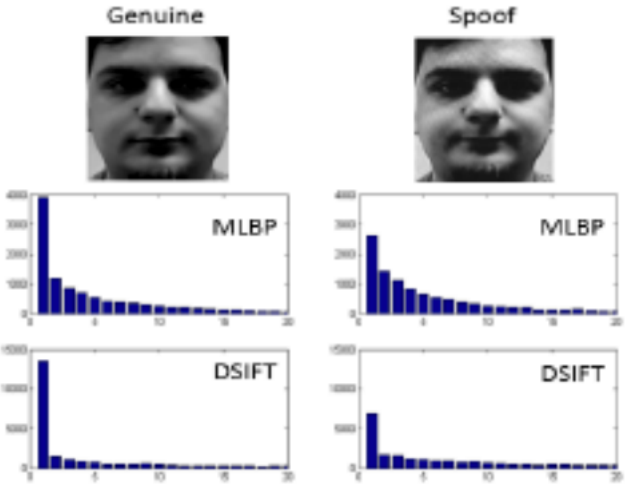
In the case of a photograph spoofing attack, visually guided hand movements are needed to orientate the photo to point in the correct direction towards the challenge. So the temporal coordination between the eye fixations and the head rotation is somehow influenced by this hand movement that it is not present in a genuine presentation.
The stimulus appears in different points on the screen and in a random sequence to prevent predictive video attacks.
Face images are then captured at each presentation of the stimulus on the screen and STASM is used to extract relevant landmarks.
Collinearity features are computed by the **Mean Square Error** between the expected positions of landmark points (given the trajectory of stimuli) and the detected ones computed.
As there are multiple face landmarks as well as several stimulus challenge trajectories, a feature vector F_{colin} can be constructed from the concatenation of these MSE values (and optionally other feature values) and used for liveness detection.
Collinearity features rather take into account the difference in landmark positions when the stimulus is presented in the same location. Even in this case a F_{coloc} vector can be computed.
One movement threshold is used to check if the attacker is trying to subvert the liveness detection system by minimizing movements of the artifact in response to the stimulus.
A second threshold is used to detect if the movements of the artifact are resulting in repeatable positioning of the eyes in response to the stimulus.
The **Optical Flow Correlation** (motion correlation) take into account the movement of the head of the user trying to authenticate and the movement of the background of the scene, because if they move together means that there is a spoofing attack, because face and background move together while they should not move together.
Optical Flow doesn't work in two cases: one case in the case of masks, but also if we have a bended face image over the attacker face.
Another possible approach is to use the Image **Distortion Analysis**. It is based on the kind of specular reflections that are produced by a printed paper surface or LCD screen (a smartphone screen for example) during capture. It is based also on image blurriness due to camera defocus. It is based on image chromaticity and contrast distortion because the real range of colors that can be captured by a natural image are much wider than the chromaticity of the imperfect color rendering and the lower chromaticity of a paper or a screen. It also based on color diversity distortion because both prints and screen images have a lower resolution with the respect to natural images also regarding the variety of colors.
There is a kind of compound analysis that is carried out according to these different elements and then it is possible to concatenate the features extracted from each of this image distortion factors and then to train the **ensemble classifier**.
What is the difference between an ensemble classifier and the previous classifiers?
In this case fusion is carried out at the end according to the separate response of each classifier.



What is important to know is that in the case of image we have a double reflection model that is applied because we have a double capture of the image and so the kind of distortion is significantly increase in the case of reproduced images.
Different spoof attacks will have different sample distributions in the IDA feature space because we may have different abnormal reflectance pattern according to the kind of paper that is used, according. to the kind of ink, etc...
In order to train system in order to detect this case of spoofing the database used contains printed images produced by different printers and captured smartphone from different smartphone.
Because regarding the color diversity feature and the chromatic features there is a difference among different visualization devices a part from the blurriness that is different from a screen to paper, but also depends to the kind of ink that is used by the print.

REPLAY ATTACK

Another way to effectively use the study of microtextures is related the **Replay Video Attacks**.
When we carry out a Replay Video Attack, we show a video on another screen. In this case there is a special kind of pattern that is called **moiré pattern aliasing** that is caused by the overlay of two pixel patterns one from the original video and the other one from the screen over which the video is shown.
It is possible to detect this special pattern using **Multi-scale LBP** and DSFIT (this tries to detect the features with the highest energy in the image).
In practice, whenever a kind of moiré pattern is present, we can conclude that we're in presence of a replay spoof attack.
The moiré pattern exists ini the entire spoof video frame, where it appears as distinct texture pattern; but whenever the presented video does not occupy the overall view, the moiré pattern can be present only in a partial region (only in the region of the recaptured video).
The authors use MLBP and SIFT for spoof detection, either individually or combined.
Each video is decoded into individual frames and each frame of the video is processed by itself.
First of all the is the detection of a face or a face region and then MLBP is applied over also DSFIT operator.
The MLBP features are calculated as: $f_{MLBP}(I) = \{LBP_{P,R}\}_{(P,R)\in\{(8,1),(24,3),(40,5)}$
The DSIFT features from each image patch are calculated using 8 orientation bins and 16 segments.



3D MASK ATTACK

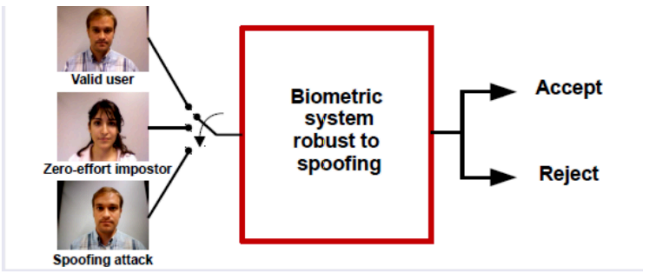
Methods that depend on the assumption of a planar surface for a fake face are rendered futile in case of 3D facial mask attacks.
In the case of 3D Mask Attack we use the reflectance characteristics of different materials that are different from the reflectance characteristics of skin. Whenever these is plastic or silicon or paper pulp, etc... we have a reflectance that is happens along different wavelengths.
So we can choose those specific wavelengths after inspecting the albedo curves of facial skin and mask materials with varying distances. Albedo is particular characteristic of the reflected light.
We inspect the albedo curves of facial skin and mask materials with varying distances because in some cases also the distance from the sensor can affect the accuracy of spoofing recognition.
There is a problem with this kind of approach: we should be able to study the albedo curves for each and any materials that could be used to build a kind of mask for spoofing. Whenever a new material appears and the new attack is carried out using it, the risk is that we cannot detect this attack.
An alternative proposal entails using a micro-texture analysis based counter measure applied separately on color images and depth maps captured from the probe sample.
A close look at the differences between masks and real faces reveals that they have **different texture and smoothness characteristics**.
Based on these observations, the LBP based approach can be used in order to detect mask attacks.
The work by Kose and Dugelay exploits exactly the same approach described for photo attack, using it both on the texture image and on the depth map.
There is also another approach that is based on the kind physiological behavior of the human body: the **photoplethysmogram**. That is a volumetric measurement of an organ. In practice, with the real face, it is possible with an advanced video capture and processing, capture micro volumetric changes that are produced by the blood flow under the subcutaneous tissues. So with very high resolution and with very high advanced processing, it is possible to study the presence of these micro volumetric changes that are produced by the blood flow.
A system that is called **FaceReader** it can be used for **Remote Diagnosis**. The speed of blood pulses can somehow measure the bit heart rate of a person. The same technology can be used for example to evaluate emotions of a person (some emotions can speed up the blood flow) and so it also possible to detect a mask, because under a mask there is no blood flow.
It is possible to learn a confidence map through heartbeat signal strength to weight local rPPG (remote PPG) correlation pattern for classification, to further exploit the characteristic of rPPG distribution on real faces.

RESEARCH OF DATASETS

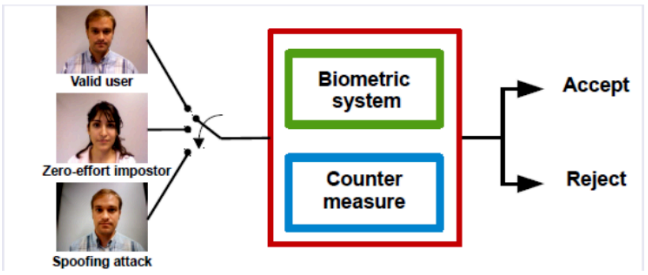
In order to study these kind of problems, **we need suitable datasets**, where there are different photo attacks that have been taken into account: move the photo horizontally, vertically, back and front; rotate the photo in depth along the vertical axis; the same as but along the horizontal axis; bend the photo inward and outward along the vertical axis; the same as but along the horizontal axis.

ANTI-SPOOFING EVALUATION

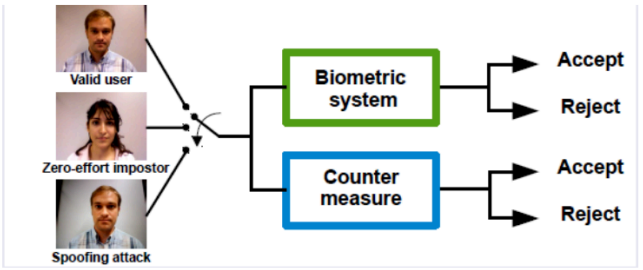
What is important to measure is the ability of the system to evaluate a possible anti-spoofing module.
As we're interested to measuring the accuracy of the system in recognized a biometric sample, we're also interested whatever is the biometric trait in evaluating the anti-spoofing ability of our system.
One possibility is to exploit the **Spoof False Acceptance Rate** (**SFAR**): the number of the spoof attacks that are falsely accepted.
When we cum to spoof detection, we've to add the Spoof False Acceptance Rate to compare it with the False Rejection Rate, because in practice False Acceptance can be due also to 0 effort attacks (and this is a problem of the classifier).
It also important to consider how many times in a spoof scenario, we've a false rejection over a genuine attempt that was misclassified as a spoof against the Spoof False Acceptance Rate.
In this case False Rejection doesn't refer to the miss-recognition of a sample, but to misclassification of the genuine sample as a spoofed attack.
In some case we can **fuse recognition and anti-spoofing** in a single system with a single Accept-Reject response. So we can combine a Biometric Classifier with a Binary Classifier, because an Antispoofing Classifier is not but a system algorithm able to decide whether a probe is genuine or not.



In the case of the Countermeasure we can also measure the **False Living Rate** (**FLR**) and the **False Fake Rate** (**FFR**), that in some sense substitute what we have defined as False Rejection Rate and Spoof False Acceptance Rate. The False Living Rate is the percentage of spoofing attacks misclassified as real, while the False Fake Rate is the percentage of real access misclassified as fake.



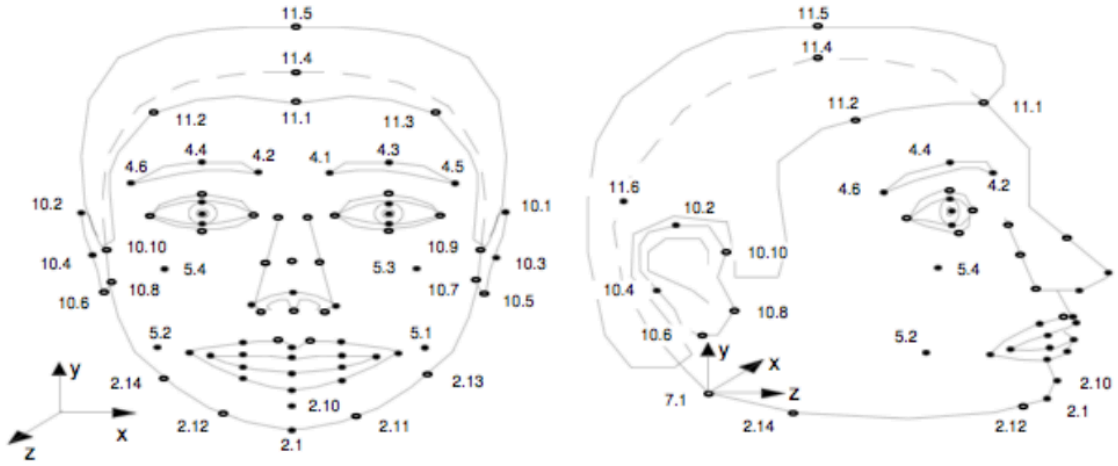
For the evaluation, we've to distinguish a **Licit Scenario** from a **Spoof Scenario**, because in the first one we have the possible errors that the system can make and so we've False Acceptance Rate (FAR) and False Rejection Rate (FRR) that can be also summarized in Half Total Error Rate (HTER) that is computed for each threshold. In particular, we will look for that specific operational point that is the Equal Error Rate (EER).



When we go to the Spoof Scenario, we still have the False Rejection Rate because we still have possible genuine users that can be rejected due a classifier error; however, in this case, we can also consider a cumulative Spoof False Acceptance Rate (SFAR). In the Licit Scenario, the False Acceptance Rate can be due by the so called zero effort attacks (a person claims an identity without taking any particular actions in order to resemble the genuine user). On the other hand, in the Spoof Scenario, we have to consider not only the zero effort attempts, but also attempts that are voluntary carried out in order to try to reproduce the appearance of the attacked person.

FACE ANTISPOOFING AT BIPlab

One of the simpler and more efficient and acceptable way for Liveness Detection (the possibility to detect whether the probe is the real probe or a print or something else like a video) is just challenging the user, so requiring a little interaction by the user at precise moment in order to detect what happens to the appearance of the probe.
The most robust spoofing detection systems in the field of face recognition rely on two main activities: the verification of face three-dimensionality; the interaction with the user.
Face three-dimensionality verification may require very sophisticated techniques.
Interaction, in most cases, involves additional hardware and software. Interaction may be modelled according to two parameters: time and content, e.g. motion type.
Requiring motion at a random time is sufficient to avoid an attack through a pre-recorded video (e.g., replay-attacks).
Challenge-response may be spoofed by video, if the system would always ask a basic and always the same head motion, e.g. turn your head from left to right.
This latter attack can be successfully addressed by requiring a specific motion type at random times, but this asks for a 3D model to track such motion and distinguish it from an appropriately presented photo.
Spoofing defence is enhanced at the expense of a significant increase of system complexity.
Poor or absent spoofing detection implies that a system can be cheated by simply showing a photo or by a video clip of a registered user.
A possible robust anti-spoofing technique relies on verifying face three-dimensionality, and on a specific user interaction.
We apply the so called **Geometrical Invariants**: are shape descriptors, that are not affected by object pose and scale, by perspective projection and intrinsic parameters of the camera. They are expressed as Ratios of distances/measures or as a combination of 3D/2D coordinates of the points of the object.
We have two kinds of Geometric Invariants. Some Geometric Invariants involve **coplanar points**, so that when those coplanar points change orientation with the respect to the capture device, such ratio should remain constant. So that they are some kind of signatures of a certain object. According to this ratio, we can infer the shape of the object or some features of the shape of the object in order to recognize it.
Another kind of Geometric Invariants is computed over **collinear points**: points that should lie on the same line and, whatever is the orientation, this kind of ratio is constant even if the line over which the points lie change position with the respect of the capture device.
All these invariants can be calculated from a single view of the object, then we can refer to them as 2D/3D invariants. They can be used as a preliminary coarse grain description of the object shape (face).
We apply these in **reverse considerations**. Given a configuration of points on an object, which are known as not coplanar, a geometric invariant which would instead require coplanarity is computed from them, on more consecutive images.
If the pose of the subject in front of the capture device changes, but the computed cross ratio stays constant, the points from which it is computed must be coplanar.
This would not be possible assuming a 3D face, and therefore the object is not 3D.
By applying this argument to face recognition, we can distinguish a real face in front of a capture device from a picture.
To add a spoofing detection-oriented interaction, the system requires to move the face, and only in those well-defined time intervals the cross ratio will be verified.



Face is not a rigid surface and to find control points which both respect the required hypotheses and that are easy to locate, turns in a difficult task.
Intuitively, the best candidates for the specific goal are those points that in a real 3D face model strongly violate collinearity/coplanarity constraints, but strictly satisfy them in a possible two-dimensional representation (photo) of the same model.
For face, good candidates are the centre of eyes, the nose tip, and the chin.
The variation v of a cross ratio c is computed over the last K frames (**observation window**) and compared with a predetermined threshold th, which is generally different for each cross ratio.
In addition, the number of frames rated as genuine (that is v > th) must be higher than a further predetermined threshold thv, which is set also according to the required level of security.
Frames which present location errors, i.e. no located faces, or incorrectly determined points, are discarded, so that they do not enter the observation window.
The number of considered frames K is a crucial parameter in terms of system performances.

FATCHA: FACE CAPTCHA

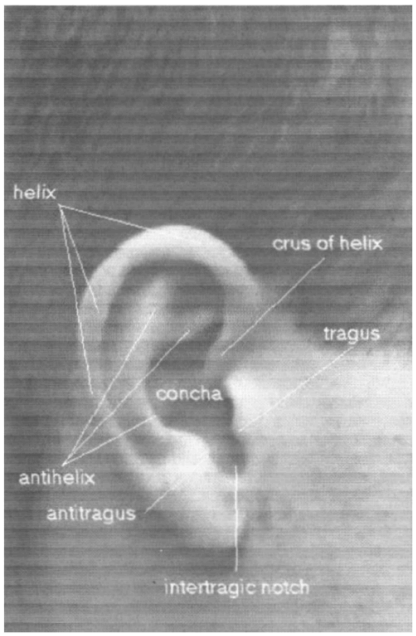
The idea is to somehow improve the usability of **CAPTCHA**.
"Completely Automated Public Turing test to tell Computers and Humans Apart" (CAPTCHA) prevents bots from using/abusing certain services and provides solution by requiring users to do something trivial for (a task which is something the user should be familiar with or a trivial task but difficult for an automatic recognition system).
We've problems with usability with the traditional CAPTCHA because for example those based on text could be hard for machine but also sometimes hard for users too.
Approaches based on images may be easer to handle, because they require much more sophisticated automatic systems but they present problems with accessibility. For example, blind people or people with low sight will never be able to solve a trivial image based CAPTCHA.
We adopted a completely different approach that is **FATCHA**. In this case, we actually require a specific gesture to the user, but a very easy one. A very easy one also to detect by the system; it's better a random combination of easy gestures that are quite easy to perform by the system. So it reduces the active role of the user to a very easy action: gesture
There is no any kind of perceptual tasks: does not involve any perceptual and cognitive task. We ask the user to produce rather than analyze something. The face of the user itself is our CAPTCHA.

The server chooses randomly from a possibly wide yet simple set of gestures and poses (limited to face and may be hand to enforce).
The server checks the user action and produces a response like in classical CAPTCHAs implementation.
Challenge is in (difficult to automatically recognize) graphical form.
The basic versions are based on images, so under sighted people still can find problems.

EAR RECOGNITION

There are several adjectives to describe a face and its components, but very few to describe an ear:
- Shape: oval, round, ...
- Appearance: delicate, plump, ...
- Facial Features: eye color, shape of the cheekbones, the shape of the nose and chin, lips thickness , …

One of the main problem is the **Occlusion** that is due by the presence of the hair. Another problem is given by the fact that **ears are inherently 3D structures**, so when we change the orientation, they need a kind of normalization procedure in order to somehow correct that change.



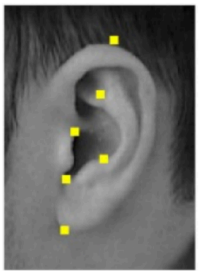
The ear has a structure that is not random, but **well-defined** just like the face :
- the external border (helix);
- the protrusion (anti-helix) running inside and parallel to the external border;
- the lobe;
- the u-shaped socket known as intertragic notch (tacca intratragica) between the entry of the inner ear (meatus) and the lobe.

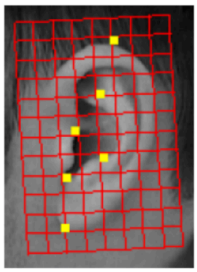
Ear is a **passive** and **static biometrics**. Since it is associated with one of human senses, it is usually uncovered for better hearing. If poorly visible it is possible to interact with the user.
The ear has been compared with other biometrics, especially the face. Some **advantages** have been underlined:
- less details, therefore requiring lower resolution;
- more uniform color distribution;
- lower if no sensitiveness to expression variations.

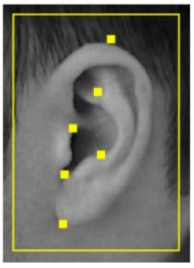
But the inherent 3D structure makes it **sensible to illumination and pose variations** and the small size is a double-sided medal.
The use of such biometrics is still relatively limited to forensics (crime scene, e.g., latent ear print analysis).
Iannarelli demonstrated that taking a certain number of landmarks on the ear and measuring distances it is possible to obtain a feature vector that is extremely discriminative, especially when we're able to capture the ear in a perfect provided condition.
Iannarelli **classified** the ear into: round, rectangle, triangle, oval. This is a first way to cut the search in order to reduce the computational time to match different templates that we have.

We can have **problem with the Localization** because it may be come a little bit difficult to dye the equal skin model of the surrounding skin.
The ear has a lower size as well as a lower complexity with respect to the face, but at the same time it has the same color of surrounding skin. This makes extraction more difficult.

There are **different approaches** in literature to ear extraction, among which: Localization of points of interest; General object detection; Geometric 3D methods.
In the case of Localization of points of interest, a possible approach uses **Neural Networks**, that must be trained. A large amount of images is acquired.





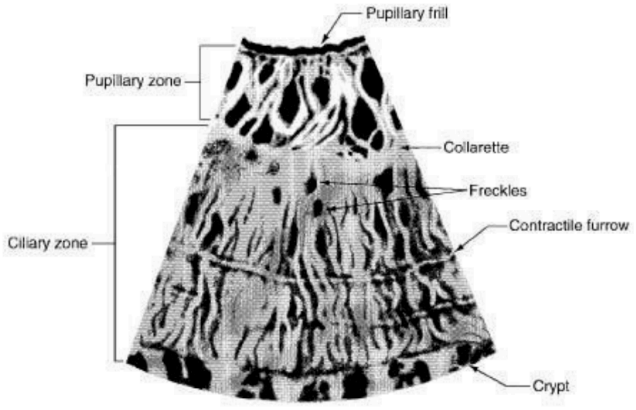


Since the training for neutral networks takes a time which is proportional to the input image size, and since in this phase we are not much interested in fine details, the images are resized to a lower resolution. The contrast in each image is normalized for better results.
The selected points determine a reference system on the image. All other points can be considered with respect to the reference ones as on a map. In this way it is possible to extract the rectangle including the ear and possible normalize its size for matching purposes.
We can apply **Object Detection** as well. There are classifiers trained to detect ears. Object Detection in fact refers to the localization of an object of a certain class within an image, in other words to the identification of its position.

AdaBoost is used here to train the model. The training is based on: multiple instances of the class of the relevant objects or positive samples; multiple negative samples, i.e. images that do not contain the object of interest.
The union of the two sets makes up the training set. During training the relevant features are extracted and selected according to their ability to classify the interesting objects.
Is it possible to add new positive or negative samples and repeat the training if respectively either the rate of missed objects or the rate of false alarms grow too much.
It is also possible to apply **3D or 2.5D approaches** because just due to its three dimensionality, the ear is well captured by range sensors.

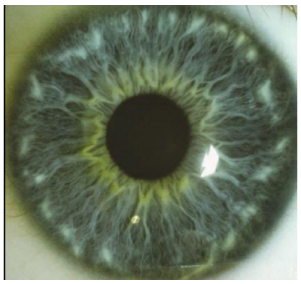
IRIS RECOGNITION

Iris has the **advantage** with the respect to the other biometrics to have to process a small and protected (a part for extreme accidents, it is very difficult to create damage on the iris) surface. Moreover it is less prone to the occlusion.
The iris is a muscle membrane of the eye, of variable color, with both shape and function of a diaphragm.
It is pigmented, located posterior to the cornea and in front of the lens, and is perforated by pupil.
It consists of a flat layer of muscle fibers which circularly surround the pupil, a thin layer of smooth muscle fibers by means of which the pupil is dilated (thereby regulating the amount of light that enters the eye) and posteriorly by two layers of epithelial pigmented cells.

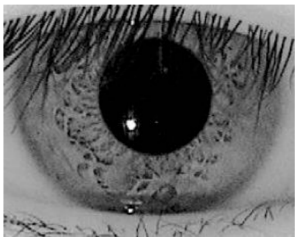


Iris colour, “regular” texture (mostly by furrows) and “irregular” patterns (e.g., freckles and crypts) **provide a very high level of discrimination**, which is comparable to fingerprints.
However, also with iris there are **problems**. The first problem is the **Reflection**. Reflection depends on the source of light and should be detected in advance. The presence of contact lens or glasses can influence the kind of reflection that can affect the iris. It is also important to consider when we have to do with a very dark iris, the texture of the iris may become completely invisible at the requested distance.
If we adopt **near to infrared sensors** we solve these two problems because it is both possible to avoid most of reflections that are founded in visible light and it is possible to highlight the texture of dark iris.
However, in the near to infrared images missing the color components. All color differences are deleted at all.
Furthermore there is another issue: near to infrared sensors are not universally available as visible light one.
Iris has a **highly randotypic components**. Randotypic means that it is not affected by genuine inheritance (there is no familiarity; no inheritance from relatives, from parents).
Which are the advantages of using iris?
The iris is **visible** yet **well protected**, so as the resolution of capture equipment increases then also the ability to capture this stricter distance increases and so we have a good advantage of using this extremely distinguishes trait.
It is a **time invariant** (after about 2 years age) and **extremely distinguishing trait** (right different from left and even twins have different irises).
Its **image** can be **acquired without direct contact**, so it is well acceptable not like the retina fundus that requires to touch the eye surface with the capture device.
It can be acquired in both near to **infrared** and **visible wavelengths**.
Which are the disadvantages of using iris?
Iris’ **surface is very limited**: only about 3.64 cm^2, so we need a very good capture device in order to obtain a good image of the iris. A “good” acquisition requires a distance of less than one meter to guarantee a sufficient resolution, depending on the input device.
Possible **problems** are represented by: the small size; the high resolution required by the equipment; the limited depth of field so that we need to pay attention to the focus and out

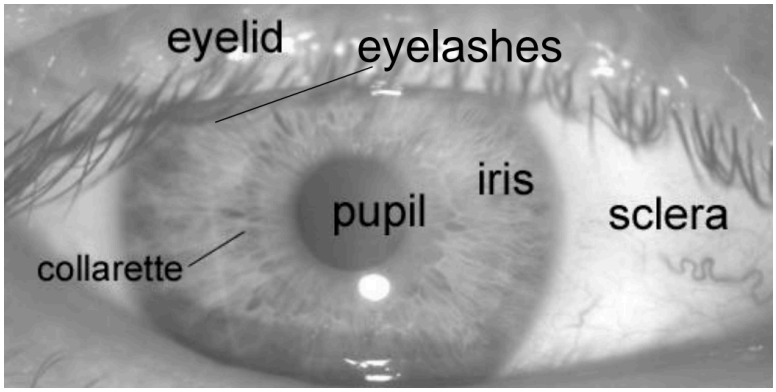
of focus problems; we need to align with optical axis unless we use some tricks like the so called "off-axis problem" (if the person is looking in another direction, the iris will not be exactly in the center of the sclera, but it will move towards the right or the left ends of the sclera); the specular reflections; the possible presence of glasses or contact lens.
We may use; CCD with high resolution; optical system that are especially designed to improve the depth of field (DOF: Depth of Field); auto-focus system according to the adaptive distance between the eye and the camera; distance sensor or estimate of the distance based on the content of the captured image; we can provide audio or visual feedback to the user; a dual-eye iris camera; pan / tilt devices to handle different heights and poses; detection and tracking of the face to guide the acquisition of the iris; infrared or near-infrared acquisition.



The **two main capture modalities** are the **Visible light** and the **Infrared light**.
In **Visible light** we have the melanin that absorbs the visible light, so we can clearly see the different colors that may appear in the iris. The layers that make up the iris are well visible. But the image contains noisy information on texture.

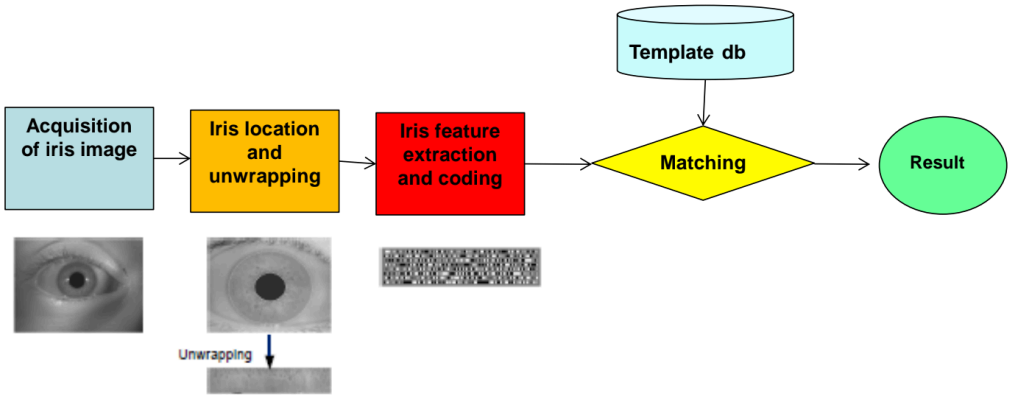


In **Infrared light** the melanin reflects most of the infrared light, so that we don't have color informations. The texture is more visible but it requires special equipment.
In the visible band of light, the iris reveals a very rich, random, interwoven texture (the "**trabecular meshwork**").
Trabecular meshwork is basically due to muscles.
In infrared illumination even dark brown eyes show a rich texture, that it could not be easy visible with Visible light.
The presence of a number of noisy elements requires a **good pre-processing/ segmentation**.

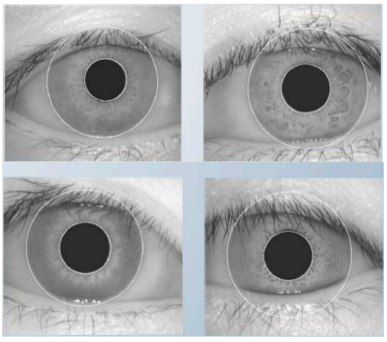


Which are the processing phases for the iris?
First of all the **Segmentation**, where we throw away the reflections and we only maintain the part that is within the circular borders that have been detected. The ideal segmentations takes segments of the full circle that only pertain to the visible part of the iris.
Then we've a **Normalization** process. Normalization is not a simple polar projection, but it is carried out to take into account the fact that in many cases the pupil is not perfectly concentric to the iris; so the pseudo-polar mapping takes into account also the points that lie on the two identified circles at regular angles and at regular distances from the center of the pupil.
We use the polar coordinates because whatever is circular in source image, becomes rectangular in the projected image, so that textural concentric circles that make up the

most important information in the iris image, become lines in this polar mapping. Lines are much more easier to process than circles from a mathematical point of view.
Then we've the **Coding phase**. This phase often takes into account the previous segmentation results, so that in general we try not to code the so called "background elements" (the don't care elements). In general, when we also carry out the same normalization process on the segmentation mask, we extract features only from the useful part of the normalized image. So we limit ourself to the regions that are classified as iris.
An iris can be processed for example by **LBP**.
At the end we've the **Matching** among two iris codes that provides a distance measure and, according to that distance and to the kind of application (either verification or identification) that we're carrying out, decides the final result of the operation.
The first and most famous **algorithm** for iris recognition is due by **Daugman**.
We've the acquisition of the iris image in strictly infrared and in strictly controlled conditions. Then we've the iris location and unwrapping. Then there is the iris features extraction with the matching with the same kind of code maintained for enrolled user.

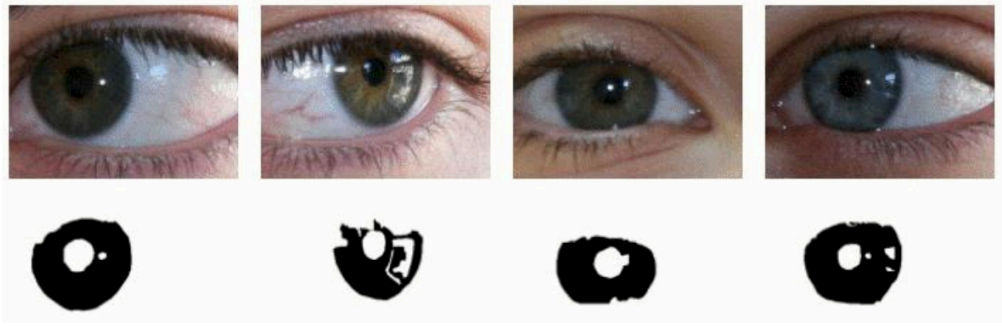


The approach uses a kind of **circular edge detector** to localize both the pupil and the iris (integer-differential operator).
The operator **exploits the convolution of the image with a Gaussian smoothing function** with center and standard deviation.
The operator looks for a circular path along which pixel variation is maximized, by varying the center r and radius (x0, y0) of a candidate circular contour.
When the candidate circle has the same radius and center of the iris, the operator should provide a peak.
In practice, it carries out a convolution and since it's using a circular function, it happens that the result of the convolution is maximized when this circular function actually finds a circular edge in the image. We repeat the convolution for each center and for each possible radius and we look for the maximum of this convolution.

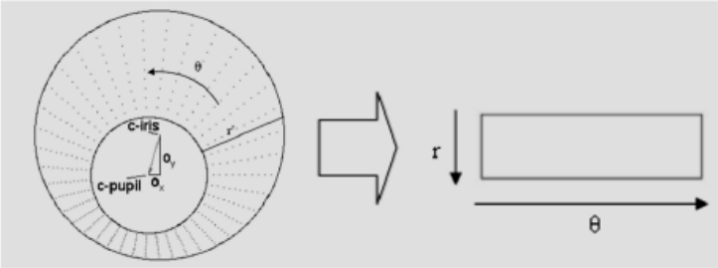


In some cases, in the **Iris Localization**, we've partial circle, but these circle produce in any case a much higher response than other circles centered in other points, because in this case we actually have a real change in the intensity in this circular part.
Daugman also carries out **Eyelids Location** because in the Iris Localization we have not only identified the iris, but also parts that should be part of the "don't care elements". Even though it falls within the iris circle, it actually represents an overlapping of an occluding reagin of the eyelids.
The procedure is similar, but instead of looking for circular paths the operator looks for arches, which are approximated by splines.

This is important because in the normalization process, we will take into account only the region belonging to the iris and so the part that still falls in the normalized region of interest will be marked as a non-iris part.
Then we pass to the **iris segmentation** and we obtain a mask so that only iris pixels are further processed.



The **iris unwrapping** is the transformation from the circular iris in cartesian coordinates into a rectangular bend in pseudo-polar coordinates.
Polar coordinates make iris processing simpler (circular bands become horizontal stripes, the overall iris annulus becomes a rectangle). Determining the right centre for the polar coordinates is of paramount important but pupil and iris are not perfectly concentric and size of the pupil can change due to illumination or pathological conditions (drunk or drugs).
Gaze direction can change the relative positions of sclera, iris and pupil. It is necessary to devise a normalization procedure: Rubber Sheet Model.



The **Rubber Sheet Model**, that was devised by Daugman, takes into account the problem of axis and especially the problem of having a pupil that is not perfectly centered within the iris.
Taking a fixed number of points on each radius that is contained between the pupil boundary and the iris boundary, it's possible to normalize the deformed distance because when we have a larger region, the sampling will be less dense. We will have a more dense sampling in the narrower region.
At the end, we will be able to obtain a representation that represents the ideal iris with the pupil perfectly centered. Since we do this for each and any pupil image, this is a kind of normalization procedure.
The model maps each iris point into polar coordinates (r, θ), with $r \in [0,1]$ and $\theta \in [0, 2\pi]$.
The center of the polar coordinates is the center of the pupil, but the transformation is not a simple polar transformation because the new coordinates for each points are a linear combination of set of points that are respectively the coordinates of the pupil contour and the those of the external iris contour.
The model compensates for pupil dilation and size variations by producing and invariant representation.
The model does not compensate for rotations. However, during matching, in polar coordinates, this is done by translating the obtained iris template until alignment.

Feature are extracted by applying Gabor filters to the I(r,θ) image in polar coordinates. Once the procedure has extracted the bend containing the iris and once that bend has been normalized, then Daugman applies Gabor filters in polar coordinates in order to obtain the so called iris code.
For each element with coordinates (r, θ) in the image I(ρ, ϕ), the method computes a pair of bits as follows:

$$h_{Re} = 1 \qquad \mathrm{Re}\left(\int\int_{\rho\phi} I(\rho,\phi) e^{-i\omega(\theta_0-\phi)} e^{-(r_0-\rho)^2/\alpha^2} e^{-(\theta_0-\phi)^2/\beta^2} \rho \; d\rho \, d\phi \right) \geq 0$$

$$h_{Re} = 0 \qquad \mathrm{Re}\left(\int\int_{\rho\,\phi} I(\rho,\phi) e^{-i\omega(\theta_0-\phi)} e^{-(r_0-\rho)^2/\alpha^2} e^{-(\theta_0-\phi)^2/\beta^2} \rho \; d\rho \, d\phi \right) < 0$$

$$h_{Im} = 1 \qquad \mathrm{Im}\left(\int\int_{\rho\phi} I(\rho,\phi) e^{-i\omega(\theta_0-\phi)} e^{-(r_0-\rho)^2/\alpha^2} e^{-(\theta_0-\phi)^2/\beta^2} \rho \; d\rho \, d\phi \right) \geq 0$$

$$h_{Im} = 0 \qquad \mathrm{Im}\left(\int\int_{\rho\phi} I(\rho,\phi) e^{-i\omega(\theta_0-\phi)} e^{-(r_0-\rho)^2/\alpha^2} e^{-(\theta_0-\phi)^2/\beta^2} \rho \; d\rho \, d\phi \right) < 0$$

Using the results of these integral computations that become kernel convolutions on digital images and according to the fact the these results are higher than 0 or lower than 0, the real parts acquire either a 1 value or a 0 value. The same happens for the imaginary part.
So that the iris code is composed by pair of values for each position in the normalized image.
We can compare two iris code using the **Hamming distance**: $HD = \frac{1}{N}\sum_{j=1}^{N} A_j \otimes B_j$
N is the total number of values in the rectangle.
For each corresponding point, we consider whether they have the same value or not. If they have the same value (whatever it is), we have a 0 in the Hamming distance; whenever the values are difference, we add a 1 in the Hamming distance. We sum up this and then divide it by the number of values.
We've also to take into account, the points that belong to the iris in both images that we're comparing. If we've different mask, one pixel that is an iris pixel in one image, cannot be an iris pixel in another image.

Matching: Hamming distance with mask

$$\mathrm{HD} = \frac{\|(codeA \otimes codeB) \cap maskA \cap maskB\|}{\|maskA \cap maskB\|}$$

FINGERPRINTS

What is the **common characteristic** that fingerprints, iris and vessels patterns share? The **prevalence of randotypic elements** in the composition of such patterns that happens during pregnancy.

A **fingerprint** usually appears as a series of dark lines that represent the high, peaking portion of the friction ridge skin, while the valleys between these ridges appears as white space and are the low, shallow portion of the friction ridge skin. Fingerprints are the traces of an impression from the friction ridges of any part of a human or other primate hand or foot. Fingerprints are easily deposited on suitable surfaces (such as glass or metal or polished stone) by the natural secretions of sweat from the eccrine glands that are present in epidermal ridges.

What is the reason why we're able to detect such prints?

When we consider the fingerprints taken by a sensor or by an ink print, they are produced by the region skin pics. On the other hand, when we consider **latent fingerprints** (the fingerprints that we leave on any suitable surface when we touch it), they are collected for example during investigations by using a special powder. In this case we've to compare a fingerprint that is usually complete (the enrolled fingerprint in the gallery) with a latent fingerprint that usually is a fraction, fragment of the complete fingerprint. So that there is the **problem of the Alignment**.

Sir Francis Galton (1822-1911), an English polymath and anthropologist, published a detailed statistical model of fingerprint analysis and identification and encouraged its use in forensic science in his book 'Finger Prints'.

Galton was the first who defined the **three basic fingerprint types** in terms of the number of '**deltas**' - discriminating whorls (2 deltas) from: loops (1 delta), and arches (no deltas). This was the first kind of classification for fingerprints.

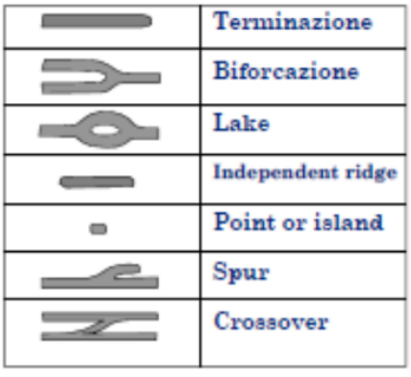
They are called **Macro-Singularities**. These Macro-Singularities were among the first features that were used in order to compare fingerprints. They are also called "**first level features**" and they entail a global consideration of the ridge pattern.

Actually, Galton was primarily interested in using fingerprints as a support to the study of racial heritage.

He also introduced the concept of **minutia** and suggested the first simple classification system, based on grouping patterns in arches, loops e whorls.

Minutiae, or **Galton features**, are micro-singularities determined by the end-points or bifurcations of ridge lines.

He discovered that even the first classification though useful in order to cut the search, it was not decisive for a final recognition. Ho noticed another kind of features that is called "**second level features**" or **Micro-Singularities**. They are special points that can be observed in fingerprints. The number of corresponding minutiae that is found in a pair of fingerprints to compare is used as a distance measure.



Another minutiae is the "core". The core is a sudden and abrupt changing direction, so it is an inner part of an arc. It's the inner most ridge in a set of riders that make up an arc.

What is the third level features?
With a very high resolution sensor it is also possible to study the pores along the ridges. They are called third level of features.

Dr. **Harold Cummins** (1894-1976) achieved world recognition as the "Father of Dermatoglyphics" or the scientific study of skin ridge patterns found on the palms of human hands.
Dermatoglyphics are all kinds of prints that can be found on palms and wherever in the body.
The findings of his lifetime studies and the techniques he developed, known as the Cummins Methodology, are accepted as important tools in tracing genetic and evolutionary relationships. The methodology has gained common usage in diagnosis of some types of mental retardation, schizophrenia, cleft palate and even heart disease.

In practice the system is rather called **Galton**-**Henry** because the two classification systems were fused together in order to make up a single classification model.
The formation of fingerprints is already complete in the seventh month of the fetal development, and the configuration of the ridges on each finger is constant during the entire life cycle.
Fingerprint ridges are formed according to a growth mode common to that of the capillary or blood vessels during angiogenesis.
Relevant factors that influence the microenvironment surrounding the fetus are the flow of the amniotic fluid and the position during the process of differentiation.
The microdiversity of the environmental conditions in the immediate vicinity of each fingertip characterizes the formation of the most minute details of their surface.
Each small microenvironmental difference is amplified by the process of cell differentiation, and there are so many changes during the formation of fingerprints to make each of them practically unique (this property is the result of empirical observations).
As the starting point of the differentiation process is the same for each footprint as determined by the same genes, the resulting patterns are not totally random.
In the case of identical twins, the minute details of the fingerprints are different, but most of the studies showed significant similarities in the classification of configurations relative to other general attributes (number, width, separation and depth of the ridges).
The **maximum difference** between the fingerprints is to be found among individuals belonging to different races. It's impossible to determine the race from the fingerprint.
We can have a kind of **ranking** of the diversity of fingerprint. They are followed in order of decreasing footprints diversity by:

- persons of the same race but without any kinship,
- father and son (who share half their genes),
- brothers and / or sisters,
- twins.

We have different kinds of acquisition. There are two basic modes of fingerprint capturing notes as **off-line** and **live-scan**.
The **off-line** acquisition takes place in **two stages**:

1. the first stage is to pass the finger on an ink pad and then leave the print on a white paper
2. then there is a a subsequent digitalization of the impression on paper through optical scanning or high resolution camera concludes the process of acquisition.

In this case the capture device may leave some kind of noises on the fingertip; also the paper that is used can also leave a special pattern. In the environment where this technique is used there is a continuous attempt to decrease the effect of these factors.
In the **live-scan or online-scan**, the digital image of the fingerprint is acquired directly through contact of the fingertip with a special sensor.
So we've a special sensor that is not a vision-based, it can be based for example on pressure. Even in this case we may have come noise due to dirt on the sensors or to other possible factors.
The **acquisition of the latent fingerprints** is done thanks special powder and special techniques that are used to transfer the fingerprint from a surface on a special paper. In this case we've a kind of process that is similar to the off-line case.
The main **parameters** that characterize the digital image of the fingerprints are:

- The **resolution** (the number of dots per inch)
- The **area of acquisition** (an area greater than or equal to a square of 1×1 inch allows the acquisition of an entire clear fingerprint)
- The **depth** (number of bits used to encode the value intensity of each pixel)
- **Contrast** (a sharp image contains better details)
- **Geometric distortion** (the maximum distortion introduced by the device or it can also be caused by an excessive pressure of the finger over the sensor surface)

The best option is to have an human operator checking the quality of the acquired fingerprint because if we have a to low pressure not all ridges are highlighted, but if the pressure if we have to much pressure there is a kind of smashing of the fingerprint.
Each kind of scanner has pros and cons. In **Optical scanner** we exploit the different optical appearance of ridges and valleys. In **Capacitive scanner** we use silicon sensors that are able to conduct electricity and according to the volume that is traveled by the electricity we can identify ridges and valleys.
A **piezoelectric sensor** uses the piezoelectric effect, to measure changes in pressure, acceleration, strain or force by converting them to an electrical charge.

OPTICAL SCANNERS

ADVANTAGES:

- they can withstand, to some degree temperature fluctuations
- they are fairly inexpensive
- they can provide resolutions up to 500 dpi.

DISADVANTAGES:

- size, the sensing plate must be of sufficient size to achieve a quality image
- residual prints from previous users can cause image degradation, as severe latent prints can cause two sets of prints to be superimposed
- the coating and CCD arrays can wear with age, reducing accuracy
- a large number of vendors of fingerprint sensing equipment are gradually shifting towards silicon-based technology

CAPACITIVE SCANNERS

ADVANTAGES:
- the silicon chip comprises of about 200*200 lines on a wafer the size of 1cm*1.5cm, thus providing a pretty good resolution for the image
- produces better image quality, with less surface area than optical
- lower cost thanks the reduced size
- miniaturization of silicon chips also makes it possible for the chips to be integrated into numerous devices

DISADVANTAGES:
- in spite of claims by manufactures that silicon is much more durable than optical, silicon's durability has yet to be proven
- with the reduction in sensor size, it's even more important to ensure that enrollment and verification are done carefully

THERMAL SCANNERS

ADVANTAGES:
- strong immunity to electrostatic discharge
- thermal imaging functions as well in extreme temperature conditions as at room temperature
- impossible to deceive with artificial fingerprint

DISADVANTAGES:
- the image disappears quickly
- when a finger is placed on the sensor, initially there is a big difference in temperature, and therefore a signal, but after a short period the image vanishes because the finger and the pixel array have reached thermal equilibrium
- this can be voided by using scanning method where the finger is scanned across the sensor which is the same width as the image to be obtained, but only a few pixel high

MINUTIAE

We can go beyond minutiae and studying the pattern of pores thanks to sensors with a sufficient resolution. In order to reliable acquire a pattern of pores we must have a sensor with the resolution in the order of 1000dpi.
At the ultra-fine observation level we can also find details intracreste, which are essentially the skin pores for sweating, whose position and shape are considered extremely distinctive.
Unfortunately, the extraction of the pores is only possible starting from the fingerprint images acquired at very high resolution, of the order of 1000 dpi, and in ideal conditions, therefore this particular representation is not practical for the majority of application contexts.

WHICH ARE THE MATCHING BASICS?

A fingerprint recognition application follows the same workflow followed by experts in the analysis of fingerprints. They take into account several factors that can be also organized into a hierarchical tree of decision, so that first of all one looks at the accord in the configuration of the global pattern, so whenever the global pattern is available we first check that the fingerprints to match have a common typology of the global pattern. This

also entails, in the case of the latent fingerprints, that we have a fragment large enough to possibly present the global patter.
In general the global pattern is quite wide enough to be also visible in latent fragments.
Then we look for a qualitative accord, that implies that the corresponding minute details are identical. In the sense that, in a certain region, it's possible to find out a bifurcation, together with an endpoint and so on and so forth.
There is also a quantitative factor because there are a minimum number of minutiae details that must match between the two prints.
There must be also a deep level of correspondence of minute details because they must be identically interrelated, in the sense that it's not sufficient that in a certain region of the fingertip we found for example two bifurcation but for example they must be divided from each other by the same number of ridges.

METHODOLOGICAL APPROACHES

There are different methodological approaches but they can be classified into **three main classes**:

- matching based on **correlation**: it is possible to compute the correlation among two images, in this case two finger images, by superimposing those and then by computing of the correlation between the corresponding pixels.
  The first problem with this method is the **Alignment**. It could not be immediate to determine, especially in automatic procedure because the matching based on correlation must be iterated for different alignment, until the best one is found.
  This kind of computation is **sensitive to non linear transformations** for example those caused by the shift of the finger when the fingerprint is captured.
  These is also a **high computational complexity**. This is mostly used for global characteristics because in practice it is a complete matching that considers all the possible aspects, in particular the macro characteristics of the fingerprints.
- matching based on **ridge features**: the extraction of minutiae in fingerprint images of low quality is problematic, therefore these methods use other features such as ridges orientation and local frequency, shape of ridges and texture, which are more reliable and easier to extract, but also less distinctive.
  This kind of method has a **low discriminating power** but it can be used as a first step in the flow in order to cut the search or to arrive more quickly to a decision.
- matching based on **minutiae**: the minutiae are first extracted from the two fingerprints and stored as two sets of points in a two dimensional space. In this case we've the problem of the **Orientation**. To find out which is the best pair of the extracted minutiae, we must take into account that, due to temporary problems with sensors, we may also have pairs of images where in one or in the other there are some lack in minutiae.
  In general the method searches for the best alignment between the two sets (the one that maximized the number of corresponding pairs of minutiae). After this we can also go into deeper details by measuring the interrelation that is if they appear in a similar pattern relating to the ridges over which they appear, their orientation, the distance between the minutiae, and so on and so forth.

Problems with fingerprint matching:

- **Scarce overlap**, for example it's possible that a finger is not well centered on the sensor and so we may have a completely different framing. We've to consider only the common part, so we try to align the common parts of the two images.
- **Different skin conditions**, for example a dry skin can hinder a correct capture of the fingerprint.
- **Non-linear distortion** due to a different pression.
- Too much movement and / or distortion.
- **Non-linear distortion of the skin** because we've a three dimensional structure that can be modify by the elasticity of skin in relation with the pressure so that if we acquire the same fingerprint in two different sessions, if we apply a different pressure they may appear. Slightly different.
- Variable Pressure and skin conditions.
- **Errors in the extraction** of the features because feature extraction algorithms are imperfect and often introduce measure errors, particularly with footprints of low quality.

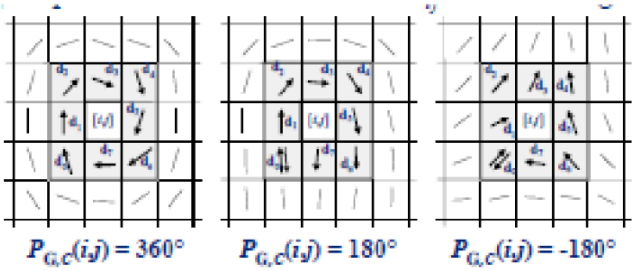
The **steps for the extraction of the feature** are: acquire the fingerprint image; then we first compute a directional map, we determine the fingerprint shape and also compute a density map; according to the directional map we can compute the singularities (the global patterns); using the directional map, the fingerprint shape and the density map, we can derive the ridge pattern and then the minutiae.
First of all we've to segment. **Segmentation** in this case indicates the separation between the foreground fingerprint (striped and oriented pattern - anisotropic) from the background which is isotropic.
What is the characteristic of this pattern? It's its **Anisotropy**, that means that is directionally dependent, because the background has identical properties in all directions while in the fingerprints we have properties of contrast and of intensity that depend on the orientation.
It's used in order to distinguish the fingerprint from the background surrounding it.
We can extract **Macro-features** from the ridges.The ridge-line flow can be effectively described by a structure called **directional map** (or **directional image**) which is a discrete matrix whose elements denote the orientation of the tangent to the ridge lines. In detail, each element [i, j], in a grid superimposed to the fingerprint image indicates the average orientation of the tangent to ridge points in a neighborhood of the point.
Analogously, the ridge line density can be synthesized by using a **density map** or **density image**, that measures the density of ridge lines in each region of the fingertip.
The local orientation of the ridge line in the position [i, j] is defined as the angle $\theta(i, j)$ that the ridge line, crossing a small at will neighborhood of point [i, j], or the tangent to the ridge lie in that point, forms with the horizontal axis.
This could be a very expensive computational process, so that many method instead of measuring orientation at each point, use a grid measure so that they measure the orientation only in fixed points in the grid.
The directional image D is a matrix in which each element corresponding to the node [i, j] of a square grid denotes the average orientation of the ridge (of the tangent to the ridge) in a neighborhood of $[x_i, y_j]$.
What is a Directional Map or a Directional Image? It is a matrix in which each element corresponding to the node [i, j] denotes the average orientation of the ridge that is tangent or the orientation in small neighborhood.
We can consider another approach that is based on the **gradient**: this is a way to extract the orientation based on the gradient of the image, but the estimation of a single orientation represents a low level analysis which is too sensitive to noise. On the other

hand it is not possible do a simple average of more gradients due to the circularity of corners. The concept of average orientation is not always well defined especially when our gird is not dense (a sparse grid). Some solutions imply doubling the angles and considering separately the averages along the two axes.
On the other hand, the **Frequency Map** relies on the number of ridges per unit land that fall along a hypothetical segment that is centered in a certain point and orthogonal to the orientation of the local ridge.
It is possible to compute an image frequency F, similar to the directional image D, by estimating the frequency in discrete locations arranged in a grid.
A possible approach is to count the average number of pixels between consecutive peaks of gray levels along the direction orthogonal to the local orientation the ridge line.

How can we find singularities?

Most approaches are based on the directional map. A practical and popular method is based on the calculation of the **Poincaré index**.

What is the Poincaré index? A directional map is vector field (a region filled of vectors with different orientations). Our fingerprint is a curve immersed in this vector field and so, the Poincaré index, is defined as the total rotation of the vectors of this field once we travel along the curve.
If G is a vector field and a curve C is immersed in G, the Poincaré index $P_{G,C}$ is defined as the total rotation of the vectors of G along C.
G is the field associated with the image of the orientations of the fingerprint image D and [i, j] is the position of the element θ_{ij} in the image.

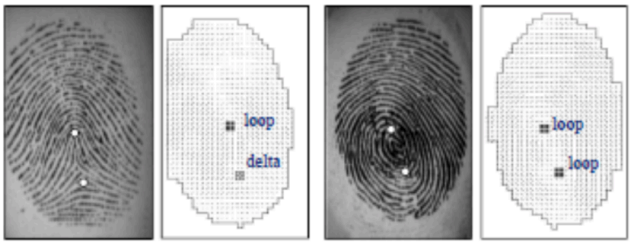


$$P_{G,C}(i,j) = \sum_{k=0..7} angle\left(\mathbf{d}_k, \mathbf{d}_{(k+1) \bmod 8}\right)$$

The index of Poincarè can be calculated as follows:

- The curve C is a closed path defined as the ordered sequence of elements of D, such that [i, j] are internal points.
- The index $P_{G,C}$ (i, j) is calculated by algebraically adding the differences in orientation between adjacent elements of C.
- The sum of the differences of orientations requires to associate a direction to each orientation. One can randomly select the direction of the first element and assign the direction closest to that of the previous element to all subsequent elements.
- It is shown that, for closed curves, the Poincare index assumes only one of the discrete values 0 °, ± 180 °, ± 360 °. In particular, regarding the singularities of fingerprints:

$$P_{G,C} = \begin{cases} 0° & \text{no singularity} \\ 360° & \text{whorl} \\ 180° & \text{loop} \\ -180° & \text{delta} \end{cases}$$



The next step is **finding up the minutiae**.
Many approaches extract minutiae and perform matching based on them or in combination with them.
In general, minutiae extraction entails:

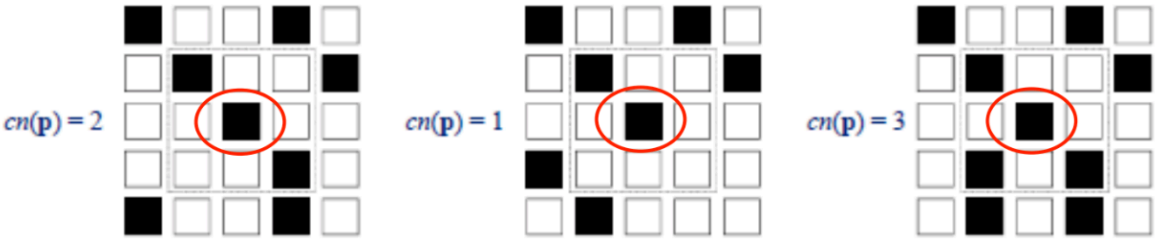
- **Binarization procedure** in order to convert a gray level image into a binary image (histogram analysis is used but it is less trivial than supposed);
- **Thinning procedure** that transforms any ridge that is thicker than 1 pixel to a 1 pixel thickness, so lines of 1 pixel.
- So once we've black and white images with a thinning of ridge traces we can proceed to scanning such ridges in order to **locate pixels that correspond to minutiae**.

Minutiae location is based on the **analysis of the crossing number**:

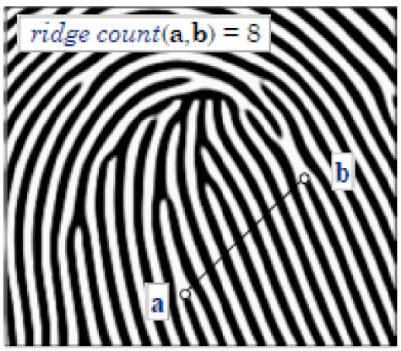
$$cn(\mathbf{p}) = \frac{1}{2}\sum_{i=1..8}\left|val(\mathbf{p}_{i \bmod 8}) - val(\mathbf{p}_{i-1})\right|$$

The crossing number is the number of changes in color that happen in the neighborhood of the pixel that is take into account in that moment.
We compute for each pixel this crossing number in order to determine whether this pixel represents a minutiae or not. There is mod8 because when we arrive at the end of the star, we've to consider the following neighbor that means to start from 1 again.
p_0, p_1, ..., p_7 are the pixels in the neighborhood of p and val(p) $\in$ {0,1} is the value of pixel p.
A pixel p with val (p) = 1:

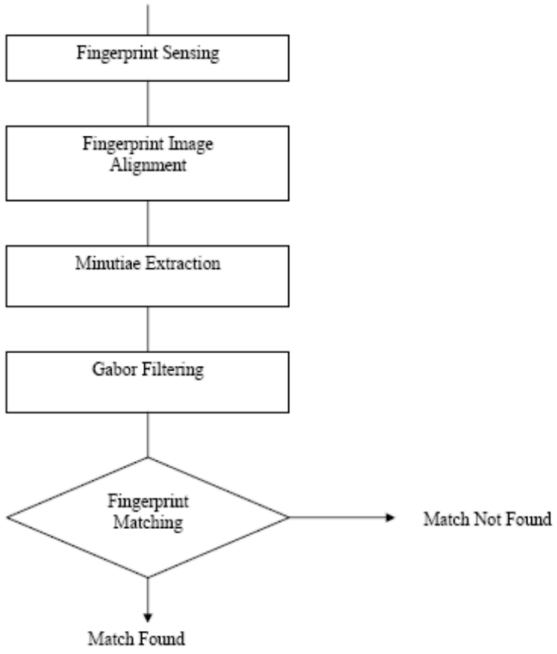
- is an **internal point** of a ridge line if cn(p) = 2, so it is a point that is not relevant for our goal because it does not represent a minutiae;
- corresponds to a **termination** if cn(p) = 1, because we have only one change in direction;
- corresponds to a **bifurcation** if cn(p) = 3;
- belongs to a **more complex minutiae** if cn(p)> 3.



It's not only necessary to find out corresponding minutiae but we also have to consider the interrelations of minutiae in two different fingerprints in order to establish if they can really be considered as correspondent.
Among the characteristics of the fingerprints used by experts there is also the **detection of the number of ridges** (**ridge count**).
The ridge count is an abstract measure of the distance between any two any points of a fingerprint. Given two points a and b of a fingerprint, the ridge count is the number of ridge lines intersected by the segment ab. The number of ridges that separate two minutiae.
Points a and b are often chosen as relevant one (e.g., core and delta).
This cannot be done for each pair of minutiae because



endpoints are not reliable enough (they can be caused by an interruption of one ridge due to bad thresholding or whatever).
In general, in order to establish the correspondence that is also useful to orientate the fingerprints, the points that over which to compute the distance are chosen are relevant ones, that is either a core (the innermost arc in a sequence of arcs) or a delta.
We can also have **Hybrid Approach** based on comparison on minutiae and texture. So we start from fingerprint sensing and fingerprint alignment, then minutiae extraction and then Gabor filtering with different frequency and different orientation of the Gabor wavelets.
At the end, we fuse the results in order to carry out the fingerprint.



We have the problem of the Image Alignment. Once we've extracted the interesting feature from a certain fingerprint, we've the problem of matching. But the problem that we meet first is the correct alignment, because if the fingerprints are not aligned any kind of comparison is not reliable.
The alignment phase starts with the extraction of minutiae from both the input and from the template to match. The two sets of minutiae are compared through an algorithm of point matching that preliminarily selects a **pair of reference minutiae** (one from each image), and then determines the number of matching of minutiae pairs using the remaining set of points.
The reference pair that produces the maximum number of matching pairs, determines the best alignment.
Thanks to the availability of information relating to the furrows for each local minutiae it is not necessary to make a comprehensive assessment of all correspondences between points.
Once the minutiae have been aligned with this method also the parameters of rotation and translation are known.
The **rotation parameter** is the average of the estimated values of rotation of all the individual pairs of corresponding minutiae, while the **translation parameters** are calculable using the spatial coordinates of the pair of reference minutiae which produced the best alignment.
The regions of the background image of the fingertip input are masked out as unnecessary. In this case we've the problem of normalizing the images.
We've **two different group of problems**: one problem is to extract relevant features from a single sample; the other problem is to find out the best way to make two samples comparable.

The firs point is the alignment, but there is also also the problem of **masking** because not all regions may be present in both fingerprints to compare and so we must also taking into account which are the corresponding regions that is better to consider.
At this point, the input images and the templates are normalized by
building on them a grid that divides them into a series of non-overlapping windows of the same size and also normalizing the light intensity of the pixels within each window with reference to a constant mean and variance.
The average distance inter-solco for a resolution of acquisition of 300x300 dpi is about 30 pixels, therefore the optimal size for each cell is 30x30 pixels.
In order to perform the **feature extraction** from the cells resulting from the tessellation a group of 8 **Gabor filters** is used, all with the same frequency, but with variable orientation. Such filtering produces 8 sorted images for each cell.
The mean absolute deviation of the intensity in each filtered cell represents its characteristic value, so one gets 8 characteristic values for each cell in the tessellation.
The characteristic values of all the cells are then concatenated into a characteristic vector.
The characteristic values relating to the masked regions in the input image are not used for the subsequent comparison phase, and are marked as missing values in the characteristic vector.
Matching can be computed considering stored templates and the characteristic vectors extracted after discarding the missing values from each template.
The comparison of the input image with the stored template is made by calculating the sum of the squared differences between corresponding characteristic vectors, after discarding the missing values.
The similarity score is combined with that obtained with the comparison of minutiae, using the rule of the sum of the combinations.
If the score of similarity is below a threshold, you can state that the input image has a corresponding template in memory and recognition is successful.

FAKE FINGERPRINTS

It is not that difficult to reproduce fake fingerprints, most of all from a cooperative user.
It's possible to create them with different materials (gelatin, silicon, latex), but each material produces fingerprints of different quality and with different characteristics either from an optical point of view or from an electrical point of view.
So depending whether we use an optical sensor or capacitative sensor or a piezoelectric sensor we may obtain a different response to a possible attack.

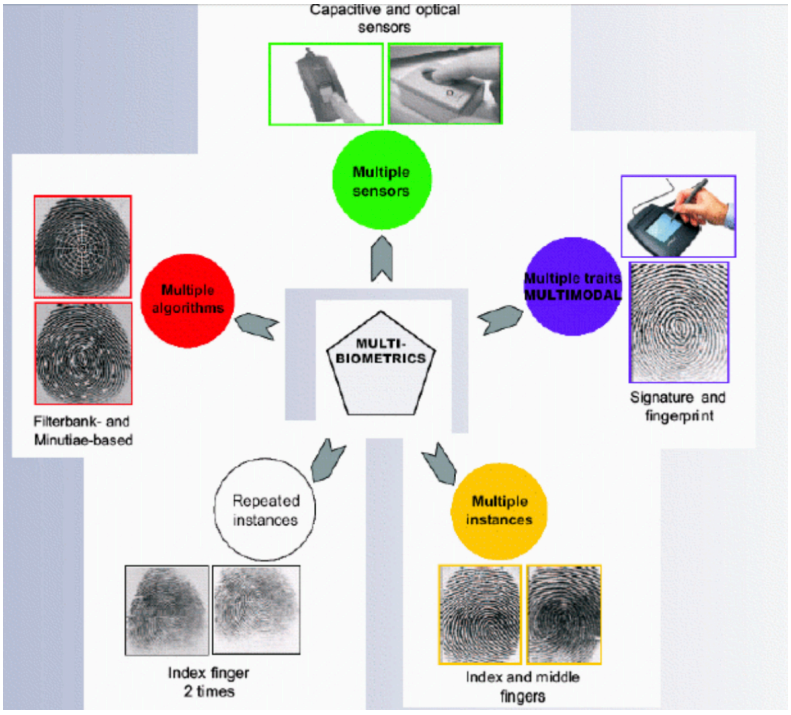
LIVENESS DETECTION

A safety measure which is particularly interesting in order to counter any attempts of cheating based on systems fingerprints, is to determine if the source of the input signal (the finger) is a living genuine biometric trait rather than a simulacrum from a fraudulent activity (a glove fitting bearing the fingerprint of an authorized user).
The logical premise of a liveliness detection test is that if the finger is alive, its print impression is actually of the person to whom it belongs.
One of the most common approaches to the vitality test consists of using one or more vital signs common to the entire population of reference, such as for example pulse and temperature.
The optical scanners of type live-scan with FTIR (Frustrated Total Internal Reflection) technology use a mechanism of differential acquisition for the ridges and furrows of

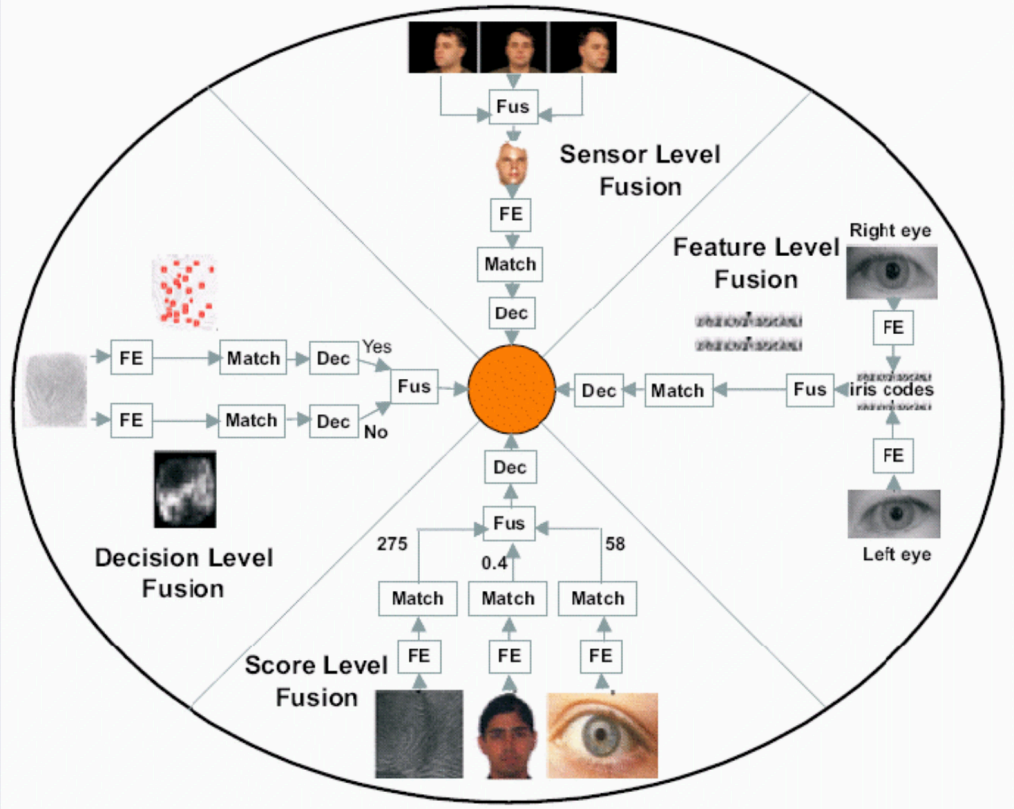
fingerprints, resulting in this way inherently more resistant to attacks by a simulacrum with two-dimensional impression of the finger.
The high resolution scan of fingerprint reveals details characteristic of the structure of the pores that are very difficult to imitate in an artificial finger.
The characteristic color of the skin of the finger changes due to the pressure when it is pressed on the scanning surface, and this effect can be detected to identify the authenticity of the finger.
Even the bloodstream and its pulsation can be detected with a careful measurement of the light reflected or transmitted through the finger.
The potential difference between two specific points of the musculature of the finger can be used to distinguish it from a dead finger.
The measurement of the complex impedance of the finger can be useful to check the vitality of the finger.
Finally, a finger sweats and this can determine the vitality of the finger.
There is a mistaken belief that the problem of digital fingerprint recognition has already been fully resolved, since it was one of the first application areas of research on pattern-recognition.
In fact many scientific and technological challenges are still to be faced to arrive to the design and implementation of a reliable system for the automatic recognition, especially in the case of fingerprints of low quality.
An example of problem that is not solved yet is the fact related to latent fingerprint the on some materials the capture of latent fingerprint may require a special process including gas, immersion or whatever.
Though valid, automatic matching and recognition systems still can not compete with the capabilities of an expert in manual techniques.
However, automated systems offer an average reliable, fast, consistent and low-cost solution to a growing number of real applications.

MULTIBIOMETRIC SYSTEMS

How can we improve the robustness of biometric system?
There are a number of limits that can be found in biometric processing (interpersonal similarity, intra-personal differences, etc..). Also there is the possibility to spoof a biometric trait like face. Face is a biometric trait that is extremely easy to spoof.
A number of countermeasures must be devised but in general in order to improve the accuracy of a biometric system, a very effective approach is to exploit **Multibiometric System**.
Multibiometric Systems or also Ensemble of Classifiers are one of the proposed solutions that can allow us to improve the performance of a biometric system whatever is the limitation.
Most present systems are based on a single biometry. This makes them vulnerable to possible attacks, and poorly robust to a number of problems.
When we use much more biometric traits is much more difficult to spoof them, because the attacker should use a fake sample for each of the biometric that is entailed in the multibiometric approach.
A multimodal system provides an effective solution, since the drawbacks of single systems can be counterbalanced thanks to the availability of more biometrics.
There is also the possibility to limit some constraint related to a single trait and also to have a better performance when we have traits that can provide a low accuracy in certain situations. We talk about traits that have more or less the same recognition power. We're considering traits that must be orthogonal in the sense that if we take the whole face and only the periocular region there is a strong correlation between these two images.
When the system includes a strong biometric trait like iris, fingerprints, blood vessels or retina vessels, it's seams useless for example use face together because the first ones are very robust by themselves; this because it's possible to find out that if we join a very strong trait and a weaker one, we may have a decrease of performance. However this can be solved with smart approaches, like providing a higher weight to the stronger biometrics.
In multibiometric systems we've more acquisition devices: one for each biometric trait. Each biometric trait can be processed individually and then, in a certain point in the process, we can merge the obtained results.
When we talk about multibiometric systems we can have many ways to combine different biometric responses. The most obvious one is to join multiple traits. This is the pure **Multimodal Approach**.
On the other hand, we can also have approaches where we can exploit **Multiple Instances** of the same trait (this is not always possible).
Another kind of multibiometric system is a system entailing **Repeated Instances**.



In this case we've exactly the same biometric trait but we capture two samples of the same biometric trait.
The **difference** with the Multiple Instances is that in this one we have for example two fingerprints of the same finger, while in the Multiple Instances we have two fingerprints of two different fingers. So in the Repeated Instances we've twice the capture of the same finger. The idea beyond the Repeated Instances is that different pression or some occasional problems can create differences in the obtained samples.
Another popular approach is the **Multiple Algorithms**. In this case we've a pure example of Ensemble Classifiers that participate in the final decision. For example we apply LBP and Wavelets to process a face image; the matching algorithms are suited to the different feature extraction methods that we apply; then we can fuse the results of two different algorithms.
At the end, we've the case of the same trait or different traits (but in general the same trait) captured by **Multiple Sensors**. For example we may use two different sensors to acquire the same fingerprint (both optical or both capacitive or one optical and one capacitive or etc...) and then fuse the results.

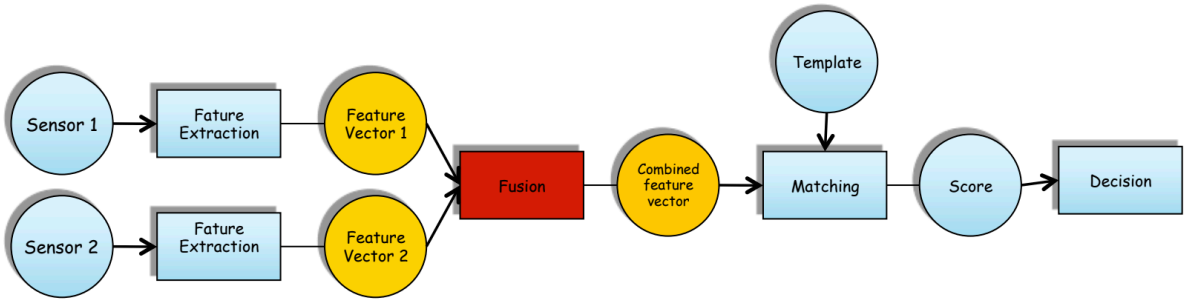


Once we have more samples for the same trait or for different traits or for different instances of the same trait, **how can we fuse the results in order to get the final response?**
The first approach is **Fusion at Sensor Level**. The Sensor Level Fusion is a very early fusion: for example when we fuse informations from three 2D images into a single 3D model, we can consider it as a kind of Sensor Level Fusion. So in this case the fusion is done before the Feature Extraction. The features that are used for matching and for decision are extracted directly from the fused model.
The Sensor Level Fusion is **difficult to achieve** because we must have first of all the same trait (in some cases it's possible to fuse different traits) and the kind of signal that is extracted must be the same.

The next point where the fusion can happen is at **Feature Level**. This means that we acquire different samples (one for each trait) and then we fuse the feature vectors that have been extracted.
An example can be the feature extraction from right eye and from the left eye; we extract separately the two iris codes and then we fuse them into a single vector that is then matched and provides the distance that underlines decision.
In this case we can also apply some different approaches; for example it's possible to extract the LBP histograms from face and to fuse them with the LBP histograms from the two irises and then apply a kind of histogram matching strategy that treats the final histogram as it was a single one.
Which are the problems entailed by these two kind of fusions?
Sensor Level Fusion requires similar kinds of signals; Feature Level Fusion may take to very huge feature vectors, so a sparse space, and so some kind of dimensionality reduction is carried out or features selection is carried out because maybe not all features extracted from all the involved samples have the same importance for recognition.
Another problem is the inflexibility of this system because the matcher is trained using the notion of the fused vector (the structure of the fused vector). Whenever we decide either to add a new sub-probe or we want to substitute one kind of probe with another kind of probe and so on and so forth, we either have to change the final classifier or to change some of its models at least.
The most popular and the most flexible approach is the **Score Level Fusion**. In this approach each trait is processed by a different subsystem. So we've a separate feature extraction, a separate matching and fusion happens after matching.
In this case the problem moves just to find a good fusion strategy.
The last possible level of fusion is represented by the **Decision Level Fusion**. In this case we're arrived to the final decision. For example, in the case of verification we may have a Yes or a No response from each of involved systems and then these are fused using a majority voting policy. So even in this case we've different ways to handle the fusion. This is the most simple from the point of view of the design implementation of all fusion approaches. The **weakness** of this approach is that whenever something goes wrong, we’ve lost all the informations that could be useful to evaluate the behaviour of the system.
The combination of the different biometries can be performed in each of the four system modules (the feature extraction module, the capture module, the recognition module and the decision module).
We can also somehow classify the possible approaches used before matching (Sensor Level or Feature Level approach) and after matching (Dynamic Selection of Features or Pure Classifier Fusion).
The **Dynamic Selection of Features** means that instead of using all responses from all the matchers, we just decide to select only a subset of the results according to some previous quality evaluation criterion.
Classifier Fusion can be carried out at either Score Level or Decision Level. Finally Score Level Fusion can be either **Abstract** or based on a **Rank** or based on a **Measurement**.

FEATURE LEVEL FUSION

Features that were extracted with possibly different techniques can be fused to create a new feature vector to represent the individual.



Better results are expected, since much more information is still present when we fuse a feature vector from one sensor or from one sample or from one biometric trait with the feature vector from another sample. In this case we may find problem with the **Compatibility**. If one of the feature extraction strategy returns an histogram and another feature extraction strategy returns a set of wavelet coefficients, Feature Level Fusion is not feasible.

Another problem that may be entailed is that we may cause "**curse of dimensionality**". A lot of features that may be fused together may create a too sparse space so at that point we can rely on some feature selections or dimensionality reduction strategies. In any case a more **complex matcher** may be required.

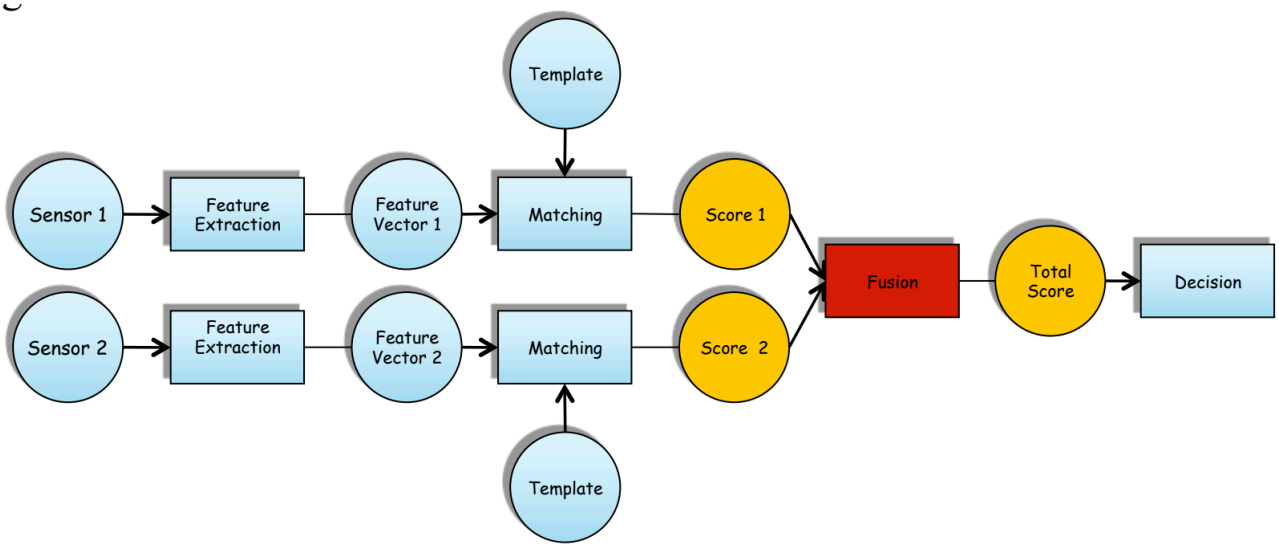
Combined vectors may include **noisy and/or redundant data**.

Feature level data fusion can be based on the simple **linking of feature vectors**.

There is also a kind of **Parallel** feature level fusion but it's limited to two biometric feature vectors. It simply uses the vector as it was a complex vector with the feature of one trait in the real part and the other feature vector was included in the imaginary part.

Another possible approach is the **Canonical Correlation Analysis** (**CCA**). The Canonical Correlation Analysis finds a pair of linear transformations, e , such to maximize the correlation coefficient between characteristics. Vectors are first reduced in dimension because CCS suffers from "Small sample size".

SCORE LEVEL FUSION

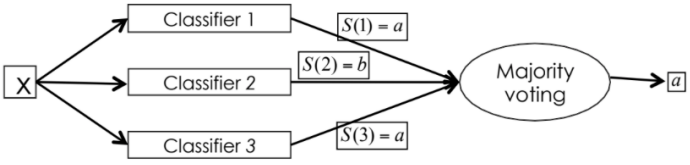


Different matching algorithms return a set of scores that are fused to generate a single final score. Each trait is completely processed by itself from the capture to the score computation.
Score Level Fusion can be carried out according to two different approaches:

- **Transformation-based:** the scores from different matchers are first normalized (transformed) in a common domain and then combined using fusion rules.
  It could be **sensible to outliers** (values that are unusually high or low just because an error of measure or just because of a very special conditions) and it may be affected by a **poor knowledge** of what is the actual minimum and the actual maximum that we can obtain from a certain measure.
- **Classifier-based**: the scores from different classifiers are considered as features and are included into a feature vector. A binary classifier is trained to discriminate between genuine and impostor score vectors (NN-Neural Networks, SVM – Support Vector Machine). Instead of trying to fuse the scores, we collect the scores into a new feature vector. Each score becomes an element in the new feature vector and then a classifier is trained with it.
  It's a complex process. We lose some grade of flexibility because if we either decide to delete or to modify or to add a new classifier, the Classifier-based approach for Score Level Fusion entails changing the score vector classifier.

What are the Fusion Rules that can be used in for the Score Level Fusion?

- **ABSTRACT**: The Abstract Level entails that the classifier just returns a class label to the input pattern. So it's neither a rank nor a score, but it is a single **class label** that represents the response of the classifier to a certain matching operation.
  Let's **divide Verification from Identification**. In case of Verification the system may return either an unknown label if the person is not recognized or the label with the claimed identity. In the case of Identification each classifier may recognize the probe as a different identity.
  One of the most way used to fuse the outcomes is to exploit **Majority Voting**: each classifier votes for a class, the pattern is assigned to the most voted class. Moreover, reliability of the multi-classifier is computed by averaging the single confidences.



- **RANK**: When we carry out Score Level Fusion we may have different situations, in the sense that the system doesn't return a single class label, but rather a **ranking of candidates**.
  We have again class labels but instead of being only one each classifier returns its own ranking.
  Each classifier outputs its class rank. In these cases each classifier can produce a class ranking according to the probability of the pattern belonging to each of them. This means that higher the probability, the higher will be the position in the list.
  Ranking are then converted in scores that are summed up; the class with the highest final score is the one chosen by the multi-classifier.

Rank	Value	C1	C2	C3
		c	a	b
		b	b	a
		d	d	c
		a	c	d

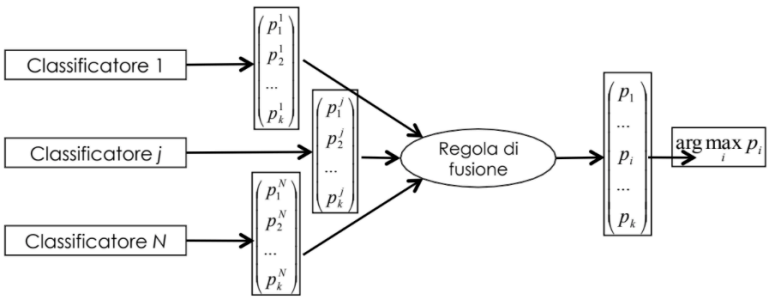
$$r_a = r_a^{(1)} + r_a^{(2)} + r_a^{(3)} = 1+4+3 = 8$$

$$r_b = r_b^{(1)} + r_b^{(2)} + r_b^{(3)} = 3+3+4 = 10$$

$$r_c = r_c^{(1)} + r_c^{(2)} + r_c^{(3)} = 4+1+2 = 7$$

$$r_d = r_d^{(1)} + r_d^{(2)} + r_d^{(3)} = 2+2+1 = 5$$

- **MEASUREMENT**: Each classifier outputs its classification score for the pattern in comparison with each class.
  The only problem we've to solve is to have values measurement that fall in the same range, so we must apply some kind of normalization.
  One trivial method is just to **sum** up the scores and to pass the result to the fusion rule.
  There is only one point that somehow occludes the full flexibility of this approach that is the possible change in the kind of threshold that we exploit to get the final result.
  Different methods are possible, including sum, weighted sum, mean, product, weighted product, max, min, ecc.



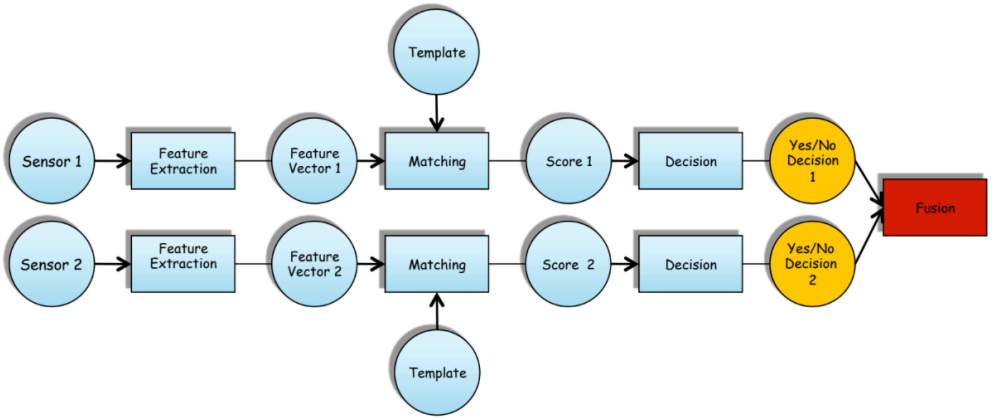
NORMALIZATION

Scores from different matchers are typically non homogeneous: Similarity/distance; Different ranges (eg. [0,1] o [0,100]); Different distributions.
To support a consistent score level fusion it is possible to exploit some score transformations (normalization), with particular attention to those laying in the overlap region between genuine and impostor.
The **issues** to consider when choosing a normalization method are:

- **Robustness**, in the sense that It must not be influenced by outliers
- **Effectiveness**, in the sense that that distribution of normalized scores must present the same shape and the same trend as the distribution of the original scores

There are a number of methods derived for this, for example the **Reliability** of the different systems can be used as a kind of weighting factor (a possible solution to reliability estimate is represented by **confidence margins**).

DECISION LEVEL FUSION



When we fuse the Decision Level, each system has already provided a decision. Each classifier outputs its decision (accept/reject for verification or identity for identification). The final decision is taken by combining the single decisions according to a fusion rule. We can use approaches that are similar to those that we've already seen.
Different combination strategies are possible. The **simplest** ones imply a **simple logical combination**:

- **Serial combination AND**: global authentication requires all positive decisions.
- **Parallel combination OR**: the user may be authenticated even by a single biometric modality.

A further important fusion rule at decision level is **Majority Voting**.
We can also, according to the results of multibiometric systems, apply a kind of **Co-Update Method**, in the sense that the so called "Highly genuine" templates that are classified by one system, can be added to the gallery together with the sample corresponding to the other trait.
The **benefit** is that if identifiers are complementary they help each other in identifying "difficult" patterns, by capturing intra-class variations in input data without lowering the acceptance threshold.
Critical Aspects of Multibiometric Systems are the normalization and also to decide which systems are more reliable than others so that their responses have to be considered with an higher weight.

QUESTIONS

1. **Haar Cascade Classifiers**: how does it works?
   A **Haar classifier**, or a Haar cascade classifier, is a machine learning object detection program that identifies objects in an image and video.
   We start with calculating **Haar Features**: weak learners based on simple (rectangular) features. They are rectangular features that can be vertically divided in light and dark or horizontally divided in light and dark or with a mixed patterns. Each feature is located in a subregion of a sub-window of the image (sliding window that we make slid on the image). Features are applied by modifying their size, shape, and position in the sub-window. The value that is returned by the system for these Haar features is the sum of the pixels values that fall under the white area minus the sum of pixels values that fall under the black area. We can establish a **threshold** for this difference so that it can somehow return a yes/no result according to the presence of this kind of pattern in the original image.
   This kind of computation can be demanding for a computational point of view. So we use the **Integral Images** to find out a way to quickly compute the value of that difference.
   We can use **AdaBoost** to select a small subset of all possible features to build a good classifier. It is a training technique that has the purpose of learning the best sequence of weak classifiers and their corresponding weights. AdaBoost learns a sequence of weak classifier and combines them by minimizing the upper bound for the classification error.
   The ideal would be minimize the outcome for each step by discarding as mach as possible non-face regions and continuing with lowest number of possible true-image regions to be submitted to the next classifier. The problem is that, once we discard a region, this will not be considered anymore and we will directly return a non-face partial result for that image region.
   Positive results from the first classifier triggers the evaluation of a second (more complex) classifier, and so on.
   A negative outcome at any point leads to the immediate rejection of the sub-window so that once we reject a certain sub-window in a certain point it will be labeled as non-face without continuing. Only the image sub-windows that successfully run the overall chain will be classified as face.
2. Why we **prefer a low False Acceptance Rate** with respect to a low False Rejection Rate?
   We mainly focus on False Acceptance when we have security related application; while the False Rejection may only be disturbing and little bit frustrating for the user, the **False Acceptance may be dangerous** in many situation because, for example, it may allow the access to a terrorist or a criminal.
   If we raise alarms for a false rejection, we can create panic, so also FR is important. In general, FA is the most critical situation but the criticality depends on the kind of applications.
3. Which is the **good rule for training**?
   When we choose the training set, **templates of different quality** must be included in the training set, because it must include as many as possible different conditions that will be found in the testing set or in the real life.
   The generalizability of outcomes, that is the possibility to get the same kind of performance even with unknown data, depends on the choice of the training set.

Dataset must **considering distortions** (PIE: Pose, Illumination, Expression; etc...) and **having good positive samples and good negative samples** (something that seems a face but it is not a face).
We may train the algorithm over a different kinds of objects or even refine the behaviour of the existing one since we can even re-train the system starting from the last results published in the available models. If we're training the algorithm in order to detect faces, we must have a huge amount of faces over which the training can be carried out. If we're looking for mouth, then we will have a huge amount of images just centered on the mouth or cropped over the mouth.
The **Positive Examples** help the system to learn which are the stereotypical elements that make up a face (in the case of face detection).
The **Negative Examples** must be at least as many as the Positive ones; these are not samples that contain the object that we're looking for but they can fool the system. So they are samples that could be wrongly classified.
We must stress the training of the system by having the system learn from the images that can fool it.

4. **Soft and strong biometric traits**. Which are the differences?
   Biometric trait is whatever pertains the appearance of a person or his behavior. Some traits are harder than others: in order to be an **hard biometric trait** that can be used whatever is our setting or with whatever kind of user, there are some characteristics that must be respected.
   An hard biometric trait is an **universal** biometric trait so that it must be owned by any person (except for rare exceptions). All people that possibly enter in the system must posses this trait. The biometric trait must be also unique. The **uniqueness** of a biometric trait means that any pair of people should be different according to that biometric trait.
   Another important requirement for a biometric trait is **permanence**: the trait must be permanent, so it mustn't change in time. **Collectability** is important because we cannot use the biometric trait, especially in a large scale, if it is hardly collectable. So the biometric trait should be measurable by some sensors.
   Lastly there is **acceptability**: involved people should not have any objection to allowing collection/measurement of the trait.
   We can have also a **soft biometric trait**: it is a trait that can be used either to cut the search or to improve the accuracy of the system; it can be useful but lacks some of such features. They are obtained relaxing the uniqueness; we can obtain some biometrics traits that while not being useful in order to recognize the single person, can be useful in order to recognize a group of person in order to reduce the search.
5. Verification systems and EER (Equal Error Rate)
   In **Face Recognition** we capture and process of user biometric data in order to render an authentication decision based on the outcome of a matching process of the stored to current template (verification 1:1 identification 1:N).
   In the **Verification** a person claims an identity and submits its sample as a probe; the system has one or more of its templates in the archive; matches this or those templates with the incoming template; before decide if that person is who he claimed to be, the system compare the similarity compared by the matching with the acceptance threshold (this threshold is decided in advance, when designing the system); if the similarity measure meets the acceptance threshold, then the claimed identity is accepted, otherwise it is rejected as an impostor. From the point of view of the identities, it is a one-to-one matches: we only match the incoming template with the single or the multiple templates in the gallery that belong to the claimed identity.
   The most **common measures to compare Verification Systems** are: plot the curve

produced by False Acceptance Rate with the curve produced by the False Rejection Rate; an important operating point that is called Equal Error Rate (ERR, it is given by that threshold where the FAR is more or less equal to the FRR); the Receiving Operating Characteristic Curve (ROC) that plots a kind of general behavior of the system (it has FAR on the x axis and 1-FRR on the y axis; higher is the curve towards the left upper half of the quadrant of the coordinates, better is the system).
The point provided by the threshold where FAR and FRR achieved a similar value is called **Equal Error Rate** (**EER**). The EER is not a threshold but it's the value that we achieved setting up a similarity threshold where the FAR is equal to the FRR.

6. How measure FA (False Acceptance) in Verification?
   When an impostor subject is accepted we've a **False Acceptance**, FA (it is also indicated as False Match, FM, or type II error). This is due to the excessive flexibility of the system that can lead to accepting the person that is not who claims to be.
   False Acceptance Rate (**FAR**) is defined as the percentage of identification instances in which false acceptance occurs. However we've to taking into account the how many imposters submitted their probes to the system; not how many probes we've processed by the system. We have two scenarios in this case: in one scenario the probe belonging to all those subjects that are not enrolled; in the other scenario the probe belonging also to the enrolled person. This because also a genuine user can declare a wrong identity.
7. FN (False Negative) and TN (True Negative). FN and TN absolute numbers are not enough, why?
   When the claimed identity is true but the subject is rejected we've a **False Rejection**, FR, or a **False Negative**, FN (it is also indicated as False Non Match, FNM, or type I error). In this case we've a genuine claim, so the person is who claims to be, but the system doesn't reach a sufficient score, so that either the similarity is to low or the distance is to high with the respect to the claimed identity.
   When an impostor subject is rejected we've a **Genuine Reject**, GR, or a **True Negative**, TN (also indicated as Genuine Non Match, GNM). The person is claiming a different identity from the true one and the system correctly provides a comparison value that make us reject the claim.
   The problem of reliability is a little bit different then considering the overall accuracy of a recognition system. Different kind of users with different trait characteristics can increase just the testing operations related to those users either False Recognition or False Acceptance. We can also have samples with low quality, for example an over-illuminated face image or a fingerprint taken by a dry fingertips. So it may happen that even with a good system, from time to time, some samples provide wrong results.
   This is why measures like **FAR, FRR, CMS, ... are not enough** to give a thorough evaluation of algorithms. First of all, because they are Ex-post measures; this means that we've anyway a ground truth then can guide our evaluation and in any case the benchmark, that is the dataset over which we carry out our testing, is extremely important for the generalizability of the result.
   For **reliable comparison** of systems we have to consider: the number and characteristics of the databases used; size of images (larger images may represent higher resolution); size of Probe and Gallery (in particular the relative size); amount and quality of addressed as well as tolerated variations; possible interoperability (e.g., cross-dataset generalization).
8. What is the **ROC curve**?
   The ROC curve is one of the most common measures to compare verification systems. The **Receiving Operating Characteristic Curve** (**ROC**) plots a kind of **general behavior of the system** (it has FAR on the x axis and 1-FRR on the y axis;

higher is the curve towards the left upper half of the quadrant of the coordinates, better is the system). It could be difficult to compare systems according to curves, so we have a single value that is valuable in order to compare the systems and it is the **area under the ROC curve**. So we compare two systems just according to the shape of that ROC curve: higher the convexity toward the left upper corner, the higher will be the area of under the curve and better will be the system. ROC depicts the probability of Genuine Accept Rate (GAR) of the system, expressed as 1-FRR, against False Accept Rate (FAR) variation. We've a range from 0 to 1; on the x axis we've the FAR and we look what happens when we've a certain False Acceptance Rate to the Genuine Acceptance.

9. Which are the differences between the Identification Closed Set and Identification Open Set?
   In the **identification** there is **no identity claim**. We take each probe and we compare them with the overall gallery. So from the point of view of the identity, we have a **one-to-N matching**, because we have a single probe in input but we have to compare it with all the probe set that we have in the gallery unless we apply something to cut the search. Identification does not entail any kind of identity claim; the system try to recognize who is the person that is trying to enter in the system. There are two types of Identification: Open Set and Closed Set.
   In the **Identification Open Set** the system determines if probe belongs to a subject in the gallery. In the open set identification, the biometric system determines first whether the individual is enrolled in the gallery. After that we want to get the identity of that person. The difference with the verification is that the individual doesn't make any identity claim. In this case we can have more possible errors situations, because they depending on the matcher and on the recognition threshold (score/similarity/distance). The **threshold** mainly **determines whether the incoming probe is recognized as a probe belonging to the gallery**. So in this case, since we've not a claimed identity but we've a matching with an entire gallery.
   Closed set identification is a special case of the open set identification where we assume that each and any probe belongs to an enrolled subject. In the identification closed set **we don't have threshold at all**. The only kind of error that we can make is to return the wrong identity, because that person will be surely in the list.
   So there is just a kind of False Rejection (when we return the right identity in a position that is not the first one) and there is no False Acceptance. Identification Closed Set is not a really realistic setting but is useful in order to evaluate the performance of the system, just because **the evaluation of Identification Open Set is more difficult to compute**.
10. How can we **calculate the performance on identification closed set**?
    We create the so called **Cumulative Match Characteristic curve** (**CMC curve**). The probability of having the correct identification within the k position is called **Cumulative Match Score at rank k (CMS)**. It therefore reports the probability that the correct identity is returned at the first place in the ordered list (CMS at rank 1), or at the first or second place (CMS at rank 2), or in general among the first k places (CMS at rank k). If the number n of ranks in the curve equals the size of the gallery, we will surely have a probability value of 1 at point n. So the Cumulative Match Characteristic curve plots the values of CMS for all possible ranks. In general, all possible ranks means that we can arrive to compute the probability that correct identity is within the end of the list so we surely reach probability 1. Since we assume that all people are in the gallery, at least we will have the correct identity as the last one. So **CMC curve always reaches the value of 1**.

Higher is the area under the curve, the better is the system. The x axis depends on the dimension of the gallery, so we can't normalize or use it in 0 and 1.
The maximum area is equal to the size of the gallery because it's always the size of the gallery multiply by 1. In order to have a normalized value, we could divide the actual area by its maximum; so we can have a value between 0 and 1.
What we're most interested in is CMS at rank 1, that means what is the probability to get the correct identity exactly in the first position. This special values is also known as **Recognition Rate**.

11. The Identification open set is more difficult than Verification?
    The **identification task is much more difficult** for biometric systems (but also for human operators) than the verification task, because we must both meet the acceptance threshold and return the correct person as the recognized identity. So that it's important to take into account the different kind of tasks to exploit the correct statistics.The performance statistics are different for different kinds of application just for this reason.
12. **Fingerprint**: which are the different levels of fingerprint recognition?
    First of all we've to segment. **Segmentation** in this case indicates the separation between the foreground fingerprint (striped and oriented pattern - anisotropic) from the background which is isotropic. The characteristic of this pattern is its **Anisotropy**, that means that is directionally dependent, because the background has identical properties in all directions while in the fingerprints we have properties of contrast and of intensity that depend on the orientation.
    Then we can **extract Macro-features from the ridges**.The ridge-line flow can be effectively described by a structure called **directional map** (or directional image) which is a discrete matrix whose elements denote the orientation of the tangent to the ridge lines. In detail, each element [i, j], in a grid superimposed to the fingerprint image indicates the average orientation of the tangent to ridge points in a neighborhood of the point.
    Analogously, the ridge line density can be synthesized by using a **density map** or density image, that measures the density of ridge lines in each region of the fingertip. The local orientation of the ridge line in the position [i, j] is defined as the angle $\theta(i, j)$ that the ridge line, crossing a small at will neighborhood of point [i, j], or the tangent to the ridge lie in that point, forms with the horizontal axis.
    We can consider another approach that is based on the **gradient**: this is a way to extract the orientation based on the gradient of the image, but the estimation of a single orientation represents a low level analysis which is too sensitive to noise. On the other hand, it is not possible to do a simple average of more gradients due to the circularity of corners. The concept of average orientation is not always well defined especially when our gird is not dense (a sparse grid). Some solutions imply doubling the angles and considering separately the averages along the two axes.
    On the other hand, the **Frequency Map** relies on the number of ridges per unit land that fall along a hypothetical segment that is centered in a certain point and orthogonal to the orientation of the local ridge. A possible approach is to count the average number of pixels between consecutive peaks of gray levels along the direction orthogonal to the local orientation the ridge line.
    Then we can find the singularities using the **Poincaré index**. A directional map is vector field (a region filled of vectors with different orientations). Our fingerprint is a curve immersed in this vector field. The Poincaré index is defined as the total rotation of the vectors of this field once we travel along the curve.
    The next step is **finding up the minutiae**. Many approaches extract minutiae and perform matching based on them or in combination with them. In general, minutiae

extraction entails: a **binarization procedure** in order to convert a gray level image into a binary image; a **thinning procedure** that transforms any ridge that is thicker than 1 pixel to a 1 pixel thickness; once we've black and white images with a thinning of ridge traces we can proceed to scanning such ridges in order to locate pixels that correspond to minutiae.

Minutiae location is based on the analysis of the **crossing number**. The crossing number is the number of changes in color that happen in the neighborhood of the pixel that is take into account in that moment. We compute for each pixel this crossing number in order to determine whether this pixel represents a minutiae or not.

13. Rubber Sheet model and when we use it.

    The **Rubber Sheet Model** takes into account the problem of axis and especially the problem of having a pupil that is not perfectly centered within the iris. Determining the right centre for the polar coordinates is of paramount important but **pupil and iris are not perfectly concentric** and **size of the pupil can change** due to illumination or pathological conditions (drunk or drugs).

    Taking a fixed number of points on each radius that is contained between the pupil boundary and the iris boundary, it's possible to **normalize the deformed distance**: when we have a larger region, the sampling will be less dense; we will have a more dense sampling in the narrower region. At the end, we will be able to obtain a representation that represents the ideal iris with the pupil perfectly centered.

    The model maps each iris point into polar coordinates where the center of the polar coordinates is the center of the pupil. The transformation is not a simple polar transformation because the new coordinates for each point are a linear combination of set of points that are respectively the coordinates of the pupil contour and the those of the external iris contour.

    The model compensates for pupil dilation and size variations by producing and invariant representation. The model does not compensate for rotations. However, during matching, in polar coordinates, this is done by translating the obtained iris template until alignment.

    Once the procedure has extracted the bend containing the iris and once that bend has been normalized, then Daugman applies **Gabor filters** in polar coordinates in order to obtain the so called iris code.

14. **Borda count** and when we use it.

    When we carry out **Score Level Fusion** we may have different situations: the system may not return a single class label, but rather a **ranking of candidates**.

    In these cases each classifier can produce a class ranking according to the probability of the pattern to belong to each of them. This means that higher the probability, the higher will be the position in the list.

    Ranking are then converted in scores that are summed up; the class with the highest final score is the one chosen by the multi-classifier.

15. What is **spoofing**? How it is carried out? Which are the strategies?

    Biometric spoofing attack is the act of fooling a biometric application, by using a copy or performing an imitation of the biometric trait that is used by that system in order to legitimately authenticate a user. In order to successfully carry out a presentation attack, it is necessary to know which is the biometric trait or which are the biometric traits underline the authentication process.

    In Biometric Spoofing we consider different kinds of attack, where the attacker actively acts in order to acquire the appearance of the attacked person.

    A biometric system can be attacked in different points: in the transmission channel; in the feature extractor; in the comparator. It’s possible to inject data over of each of such channel in order to force the result that we want. It is also possible to

attack the Database with templates that don't belong to enrolled people. Especially when there is an automatic update of the gallery, in order to address the change of appearance of a person, it is possible to "poison" the sub-gallery for that subject by injecting images.
All of these kinds of attacks along the channels, the database and so on and so forth, can be considered as **Indirect Attacks**. Defending from these kinds of attacks is a matter of cybersecurity.
Another kind of attacks are the **Direct Attacks**. In this case, our final aim is not to recognize the person, but rather to detect the attack so that we can proceed with a countermeasure.
We can do a Classification of spoofing attacks: 2D spoofs attack and 3D spoofs attack.
In the **2D Spoofs** the attack is carried out by using a 2D surface like a photo or like a video that is played once we have somehow recorded the attacked person. **Photo attacks** can use either hard copy, hard copy with a eye whole, screen. An alternative for 2D spoof is to exploit **Video Attacks**. In that case we recorded the person that we want to attack using different screen resolutions and then present this video instead of our true face. This is also called "**Replay Attack**".
In the **3D Spoofs** the attack requires that there is not a plain surface but a three dimensional surface and in general masks are used for this kind of attack (**Mask attacks**).
For a human operator, it's trivial to detect such kind of spoof attack; especially when the image that is shown to the system does not occupy the overall screen.
An automatic system is designed in order to identify region of interest containing face; since in this case the region of interest containing the face is present, there are all the things which automatic system cares about. Unless we extend the functionality of the system with an anti-spoofing procedure, it is quite easy to carry out this kind of attack.
For each kind of attack there are different **anti-spoofing techniques** that are mostly based on **liveness detection**. In general, what we try to do, a part from studying the reflection pattern of the surface and the microtexture, is to "challenge" the user in order to detect the kind of reaction of the face that we've in front.
We've anti-spoofing techniques that work at **Sensor-level** (**Hardware Based**); in practice they exploit the intrinsic properties of a living body including the kind of reflectance that is producing and involuntary signals (for example micr movements of the eyes) that are completely absent in a photo or in a mask. An example of anti-spoofing technique that takes advantage of these kind of movements is **Eye Blink**. In fact, the eye blinking is a natural involuntary movement that can be detected in order to distinguish a real person from a photo.
An the other hand, we can have a voluntary response from the so called **“Challenge-Response” strategy**. In this case we can ask to a person that is presenting a probe to make a certain kind of expression or movement. If what we expected doesn’t happen on the face that is shown to the system, we can conclude that this is a spoof attack.
Other kinds of anti-spoofing techniques are carried out at **Feature Extractor level** (**Software based**) and they can be either static or dynamic.
An example of study static is the study of the microtexture of the acquired image; with dynamic we study the kind of dynamic features that a certain movement can produce when it is naturally carried out.
We can have also a **Score level fusion** where we can either fuse anti-spoofing and

recognition in a single module or carry out anti-spoofing in advance and proceed with the recognition only if the anti-spoofing returns a genuine response.

16. How **evaluate the spoofing**? Is it possible to evaluate it together the recognition? How?
    One possibility is to exploit the **Spoof False Acceptance Rate** (**SFAR**): the number of the spoof attacks that are falsely accepted.
    When we cum to spoof detection, we've to add the Spoof False Acceptance Rate to compare it with the False Rejection Rate.
    In this case **False Rejection** doesn't refer to the miss-recognition of a sample, but to misclassification of the genuine sample as a spoofed attack.
    For the evaluation, we've to distinguish a **Licit Scenario** from a **Spoof Scenario**, because in the first one we have the possible errors that the system can make and so we've False Acceptance Rate (FAR) and False Rejection Rate (FRR) that can be also summarized in Half Total Error Rate (HTER) that is computed for each threshold. In particular, we will look for that specific operational point that is the Equal Error Rate (EER).
    When we go to the **Spoof Scenario**, we still have the False Rejection Rate because we still have possible genuine users that can be rejected due a classifier error; however, in this case, we can also consider a cumulative Spoof False Acceptance Rate (SFAR).
    In the Licit Scenario, the False Acceptance Rate can be due by the so called **zero effort attacks** (a person claims an identity without taking any particular actions in order to resemble the genuine user). On the other hand, in the Spoof Scenario, we have to consider not only the zero effort attempts, but also attempts that are voluntary carried out in order to try to reproduce the appearance of the attacked person.
    In some case **we can fuse recognition and anti-spoofing** in a single system with a single Accept-Reject response. So we can combine a Biometric Classifier with a Binary Classifier, because an Antispoofing Classifier is not but a system algorithm able to decide whether a probe is genuine or not.
    In the case of the **Countermeasure** we can also measure the **False Living Rate** (**FLR**) and the **False Fake Rate** (**FFR**), that in some sense substitute what we have defined as False Rejection Rate and Spoof False Acceptance Rate. The False Living Rate is the percentage of spoofing attacks misclassified as real, while the False Fake Rate is the percentage of real access misclassified as fake.
17. Traits that lack uniqueness and permanence.
    The **Soft Biometric Traits** either lack uniqueness (e.g., hair color) or persistence (e.g., behavioural that are affected by mood, health, etc.) but can be used to limit the search.
18. Why is the iris so unique?
    Iris has a **highly randotypic components**. Randotypic means that it is not affected by genuine inheritance (there is no familiarity; no inheritance from relatives, from parents). So iris is a extremely distinguishing trait (right different from left and even twins have different irises).
19. Which are the levels of **classification in the fingerprint**?
    Galton was the first who defined the three basic fingerprint types in terms of the number of '**deltas**' - discriminating whorls (2 deltas) from: loops (1 delta), and arches (no deltas). This was the first kind of classification for fingerprints. They are called **Macro-Singularities**. These Macro-Singularities were among the first features that were used in order to compare fingerprints. They are also called "**first level features**" and they entail a global consideration of the ridge pattern.

Galton also introduced the concept of **minutia** and suggested the first simple classification system, based on grouping patterns in arches, loops e whorls.
Minutiae, or Galton features, are **micro-singularities** determined by the end-points or bifurcations of ridge lines.
He discovered that even the first classification though useful in order to cut the search, it was not decisive for a final recognition. He noticed another kind of features that is called "**second level features**" or Micro-Singularities. They are special points that can be observed in fingerprints. The number of corresponding minutiae that is found in a pair of fingerprints to compare is used as a distance measure.
With a very high resolution sensor it is also possible to study the **pores** along the ridges. They are called "**third level of features**".

20. What is a Latent Fingerprint?
    The **Latent Fingerprints** are fingerprints that we leave on any suitable surface when we touch it; they are collected for example during investigations by using a special powder. In this case we've to compare a fingerprint that is usually complete (the enrolled fingerprint in the gallery) with a latent fingerprint that usually is a fraction (fragment of the complete fingerprint).
    The acquisition of the latent fingerprints is done thanks special powder and special techniques that are used to transfer the fingerprint from a surface on a special paper.
21. Which is the problem to match a latent fingerprint with a registered fingerprint?
    In a matching between a latent fingerprint with a registered fingerprint there may be the problem of the **Alignment**.
22. What is a **multibiometric approach**?
    There are a number of limits that can be found in biometric processing (interpersonal similarity, intra-personal differences, etc..). Also there is the possibility to spoof a biometric trait like face. A number of countermeasures must be devised but in general in order to improve the accuracy of a biometric system, a very effective approach is to exploit Multibiometric System.
    Multibiometric Systems or also **Ensemble of Classifiers** are one of the proposed solutions that can allow us to improve the performance of a biometric system whatever is the limitation. When we use much more biometric traits is much more difficult to spoof them, because the attacker should use a fake sample for each of the biometric that is entailed in the multibiometric approach.
    A multimodal system provides an effective solution, since the drawbacks of single systems can be counterbalanced thanks to the availability of more biometrics.
    There is also the possibility to limit some constraint related to a single trait and also to have a better performance when we have traits that can provide a low accuracy in certain situations. We talk about traits that have more or less the same recognition power because if we join a very strong trait and a weaker one, we may have a decrease of performance.
    When we talk about multibiometric systems we can have many ways to combine different biometric responses. The most obvious one is to join multiple traits. This is the pure **Multimodal Approach**.
    On the other hand, we can also have approaches where we can exploit **Multiple Instances** of the same trait (this is not always possible).
    Another kind of multibiometric system is a system entailing **Repeated Instances**. In this case we've exactly the same biometric trait but we capture two samples of the same biometric trait.
    Another popular approach is the **Multiple Algorithms**. For example we apply LBP and Wavelets to process a face image; the matching algorithms are suited to the different feature extraction methods that we apply; then we can fuse the results of two different

algorithms.
At the end, we've the case of the same trait or different traits (but in general the same trait) captured by **Multiple Sensors**. For example we may use two different sensors to acquire the same fingerprint (both optical or both capacitive or one optical and one capacitive or etc...) and then fuse the results.

23. How can we **fuse the results in multibiometric approaches**?
    The first approach is **Sensor Level Fusion**. It is a very early fusion: for example when we fuse informations from three 2D images into a single 3D model, we can consider it as a kind of Sensor Level Fusion. So in this case the fusion is done **before the Feature Extraction**. The features that are used for matching and for decision are extracted directly from the fused model.
    The Sensor Level Fusion is difficult to achieve because we must have first of all the **same trait** (in some cases it's possible to fuse different traits) and **the kind of signal that is extracted must be the same**.
    The next point is the **Feature Level Fusion**. We acquire different samples (one for each trait) and then we fuse the feature vectors that have been extracted. An example can be the feature extraction from right eye and from the left eye; we extract separately the two iris codes and then we fuse them into a single vector that is then matched and provides the distance that underlines decision. In this case we can also apply some different approaches; for example it's possible to extract the LBP histograms from face and to fuse them with the LBP histograms from the two irises and then apply a kind of histogram matching strategy that treats the final histogram as it was a single one.
    The Feature Level Fusion may take to very **huge feature vectors** (“curse of dimensionality”, a sparse space) and so some kind of dimensionality reduction or features selection is carried out because maybe not all features extracted from all the involved samples have the same importance for recognition.
    Another problem is the **inflexibility** of this system because the matcher is trained using the notion of the fused vector (the structure of the fused vector). Whenever we decide either to add a new sub-probe or we want to substitute one kind of probe with another kind of probe and so on and so forth, we either have to change the final classifier or to change some of its models at least.
    The most popular and the most flexible approach is the **Score Level Fusion**. In this approach each trait is processed by a different subsystem. So we've a separate feature extraction, a separate matching and the fusion happens after matching.
    In this case the problem moves just to **find a good fusion strategy**.
    The last possible level of fusion is represented by the **Decision Level Fusion**. In this case we're arrived to the final decision. For example, in the case of verification we may have a Yes or a No response from each of involved systems and then these are fused using a majority voting policy. So even in this case we've different ways to handle the fusion. The **weakness** of this approach is that whenever something goes wrong, we’ve lost all the informations that could be useful to evaluate the behaviour of the system.
24. What is the **Detection and identification rate**?
    In the **Identification Open Set**, when we have a correct detection, we can also consider the position in the list where the first template for the correct identity is returned (because the correct identity may not be in the first position). So we can consider its rank, that is the position in the list of the template belonging to the correct identity.

Detection and Identification Rate (DIR) at rank k, measure the the probability of a correct identification at rank k (the correct subject is returned at position k).
The probability of false reject expressed as 1-DIR(at rank 1), so FRR(t) = 1 - DIR(t, 1).
The Detection and Identification Rate at rank 1 is more or less equivalent to Genuine Acceptance; False Rejection is the complement of the GAR, so for this reason we can compute FRR just doing: 1 - DIR(at rank 1).

25. What's missing from **behavioral biometric traits**?
    Behavioral traits are difficult to reproduce but suffer from **permanence**. Examples of behavioral features are signature biometrics and key typing.
26. **Doddington zoo**. Why was the extension done?
    Most errors in the system can be attributed to specific class of users. The fact that a person belongs to one of the bad classes of Doddington zoo may affect the reliability of the single recognition operation. Doddington defined Sheep, Goats, Lambs and Wolves in the context of speaker recognition systems.
    A **Sheep** is a person that he produces a biometric sample that matches well to other biometric samples of the same person and poorly to those of other people. As such, Sheep generate fewer false accepts and rejects than average (Normal average behaviour). This is the only good class because a person who is a Sheep will mostly achieve Genuine Acceptance and seldom achieve a False Acceptance when it claims a different identity.
    A **Goat** is a person that produces a biometric sample that poorly matches to the other biometric samples of itself. These low match scores imply a higher than average false reject rate for Goats. This is a person that achieves higher than average FRR because this person is not well recognized.
    A **Lamb** is a person that can be easily impersonated. When the biometric sample of such a person is paired to a biometric sample from a different person, the resulting match score will be higher than average. Consequently, false matches are more likely. This may happen for example for children.
    A **Wolf** is a person that is good at impersonation. When such a person presents a biometric sample for comparison they have an above average chance of generating a higher than average match score when compared to a stored biometric of a different person. So we have a lot of False Acceptances.
    Goats, lambs, and wolves are defined in terms of a user's average genuine or imposter scores. If we consider both kind of errors, we may increase the number of classes.
    New additions to the biometric menagerie by Yager and Dunstone are defined in terms of both a user's genuine and imposter scores.
    Chameleons always appears similar to others. That is something slightly different from Wolves: in practice, Chameleon have high both average score with it compared with itself and high average score when it plays the role of impostor. So they receive high match score for all verifications. For this reason, Chameleons rarely cause False Rejects, but are likely to cause False Accepts. An example of a user who may be a Chameleon is someone who has very generic features that are weighted heavily by the matching algorithm.
    Phantoms lead to low match scores regardless of who they are being matched against. So Phantoms have a low average genuine score and also a low average impostor score. Phantoms may be the cause of False Rejections but unlikely to be involved in False Accepts. A potential cause for Phantoms would be a group of people who have trouble enrolling in the system. This would lead to feature extraction difficulties and consequently low match scores for all verification.

For example a person with a poor skin on the fingerprints can have problems with automatic enrolling.

Doves have the high genuine average score and lower average impostor scores, so this is something even better than Sheeps. They are pure and recognizable, matching well against themselves and poorly against others. Doves are rarely involved in any type of verification error. In a biometric system, Doves may be users who have an uncommon characteristic (for example a very distinctive nose). This would lead to both high genuine match scores and low impostor scores.

Worms are the worst conceivable users because they have a low genuine average score but a high average impostor score. They have few distinguish characteristics so few are correctly recognized as themselves but they can successfully impersonate other people.

27. Difference between spoofing and camouflage.
    We have to also distinguish **Biometric Spoofing** from **Camouflage** or **Disguise**.
    In the case of Camouflage the attack is carried out presenting an artifact biometric trait to the system to fool it pretending not to be oneself. So in that case **we don't want to be recognized**. If we introduce extraneous elements on the face we may have that the detection process fails.
28. Are all biometric traits subject to spoofing?
    Those who are based on appearance do. Gait, key typing, the way you sign, etc., no.
29. **Anti-spoofing on the face**
    One of the most intuitive approaches to 2D Print Attach is **Liveness Detection**. In practice, the essential difference between the live face and photograph is that a live face is a fully three dimensional object while a photograph could be considered as a two dimensional planar structure. This means that we can use structure from motion to derive kind of depth information to distinguish a live person from a still photo.
    The **disadvantages** of depth information are: it is hard to estimate depth information when head is still (we have to ask for a movement, otherwise it is very difficult to carry out a liveness detection in this trivial way) and the estimate is very sensitive to noise and lighting (if we're in a dark environment, this may help the attacker to hide some feature change with the respect to a real face).
    It is possible to compute the **optical flow** (this is a technique that tries to extract the motion vector by comparing the position of each pixel in one frame and in the following one) on the input video to obtain the information of face motion for liveness judgment, but it is **vulnerable** to photo motion in depth and photo bending.
    A possible **multimodal approach** fuses face-voice against spoofing exploiting the lip movement during speech.
    Among the earliest approaches, it is possible to consider those relying on eye blinking analysis because it relies on the natural pattern that is produced on a regular basis in eye dynamic. **Eyeblink** is a physiological activity of rapid closing and opening of the eyelid and it's a movement that cannot be avoid at all.
    An eyeblink activity can be represented by an image sequence consisting of images. The typical eye states are opening and closing. In addition, there is an ambiguous state when blinking from open state to close or from close state to open. It is possible to define a three-state set for eyes, $Q = \{\alpha : \text{open}, \gamma : \text{close}, \beta : \text{ambiguous}\}$. A typical blink activity can be described as a state change pattern of $\alpha \rightarrow \beta \rightarrow \gamma \rightarrow \beta \rightarrow \alpha$.
    An other possibility is based on **Micro-texture analysis**. This is especially efficient when we have 2D Print attacks or Photo attacks because any kind of photographic paper or printing paper has a kind of micro-texture that can be not visible by eyes but can be detected by using texture features.
    Human faces and prints reflect light in different ways because a human face is a

complex non rigid 3D object (so it reflects light in different directions) whereas a photograph is a planar rigid object (different specular reflections and shades). The surface properties of real faces and prints, e.g. pigments, are also different because natural pigments are different from ink pigments. These last ones contain some metallic components so this affects the way light is reflected. The work exploits **multi-scale local binary patterns** (LBP). The strategy that is used in multi-scale local binary patterns is more or less the same but much more simpler to adopt than the bank of wavelets with different frequencies. Multi-scale local binary patterns are computed by using windows of different size. As a further advantage, the same texture features that are used for spoofing detection can also be used for face recognition. The vectors in the feature space are then fed to an **SVM classifier** which determines whether the micro-texture patterns characterize a live person or a fake image.
Another important approach is the **Captured-Recaptured approach**. We must consider that all the distortions that are present when we capture an image affect the face image when it passes through the camera system. Such distortions are applied twice when we take the photo of a person and then when we print it. In practice, when we use a photo taken by a photo, like it may happen when using a photo taken on the internet, we have a much lower image quality compered to the capture of a real face image.
Another approach is the **Gaze Stability**. There is an algorithm proposed by Ali et al. that is based on the assumption that the spatial and temporal coordination of the movements of eye, head and (possibly) hand involved in the task of following of a visual stimulus is significantly different when a genuine attempt is made compared with certain types of spoof attempts.
In a **challenge response strategy**, when the system asks us or to the genuine subject to rotate the head, the kind of temporal coordination between the movement of the eyes, the movement of the head (if they are possibly involved) is different when if the person is looking at the screen through the holes in a mask.
The task, or the so called "challenge", requires head/eye fixations on a simple target that appears on a screen in front of the user.
In the case of a **photograph spoofing attack**, visually guided hand movements are needed to orientate the photo to point in the correct direction towards the challenge. So the temporal coordination between the eye fixations and the head rotation is somehow influenced by this hand movement that it is not present in a genuine presentation. The **Optical Flow Correlation** (motion correlation) takes into account the movement of the head of the user trying to authenticate and the movement of the background of the scene, because if they move together means that there is a spoofing attack, because face and background move together while they should not move together. Optical Flow **doesn't work in two cases**: one case in the case of masks, but also if we have a bended face image over the attacker face.
Another possible approach is to use the Image **Distortion Analysis**. It is based on the kind of specular reflections that are produced by a printed paper surface or LCD screen (a smartphone screen for example) during capture. There is a kind of compound analysis that is carried out according to these different elements and then it is possible to concatenate the features extracted from each of this image distortion factors and then to train the ensemble classifier.
Another way to effectively use the study of **microtextures** is related the **Replay Video Attacks**. When we carry out a Replay Video Attack, we show a video on another screen. In this case there is a special kind of pattern that is called **moiré pattern aliasing** that is caused by the overlay of two pixel patterns: one from the original video and the other one from the screen over which the video is shown.

It is possible to detect this special pattern using **Multi-scale LBP** and **DSFIT** (this tries to detect the features with the highest energy in the image). In practice, whenever a kind of moiré pattern is present, we can conclude that we're in presence of a replay spoof attack.

30. Difference in iris recognition: difference between infrared light and visible light.
    The two main capture modalities are the Visible light and the Infrared light.
    In **Visible light** we have the melanin that absorbs the visible light, so we can clearly see the different colors that may appear in the iris. The layers that make up the iris are well visible but the image contains noisy information on texture.
    In **Infrared light** the melanin reflects most of the infrared light, so that **we don't have color informations**. The texture is more visible but it requires special equipment.
31. Advantages and disadvantages of **3D and 2D recognition**
    Regarding the **Pose variations**, while 2D recognition can be affected by rotation, pitch and yaw variations in one pose, this is not true with 3D. When we carry out a recognition strategy based on 3D we actually don't use a 2D/flat image but we use a 3D model and so we can test with different poses of this 3D model because it can be rotated and its pose can be also changed according to pitch and yaw so that we can try to have the 3D model assume the same pose represented in the 2D image.
    For a similar reason also **Illumination** problems can be considered as solved because, as in the same way, we can rotate the model and also synthetize different illuminations.
    While **Expression** still affects face recognition in 3D. If we think especially on exaggerate expressions that are the ones more difficult to capture even in 2D, they still create problems in 3D because they can cause a real modification with the respect to a neutral 3D model.
    Also **Aging** can affect 3D strategy. According to an increasing age, it's possible that some face issues relax. Since the volume of a certain anatomic part can increase with age, this affects 3D strategy too.
    Makeup doesn't affect 3D model at all because only considering the geometric relationships, the 3D volume determined by the face doesn't change.
    Plastic Surgery can affect 3D models according to the kind of weight and the extension of the intervention.
    Occlusion affects 3D as it affects 2D, because when we build the model for the probe and it wears a scarf or glasses or whatever, this model is poorly matched against the enrolled one.
    We can summarize the **pros and cons of 3D strategies** by saying that: in 3D we have much **more information**, the built models are much **more robust** to a number of distortions, there is the possibility to **synthetize** (approximate) **2D images** from virtual 3D poses and expressions computed from a 3D model (**PROs**); the **cost of devices**, **computational cost** of procedures, **possible risk** that some acquisition devices can present, for example the laser scanner is dangerous for the eye (**CONs**).
32. **Morphable model**
    Expression still affects face recognition in 3D. It's possible to exploit the so called **morphable model**, that can be somehow distorted in order to reproduce any kind of expression. But the computational complexity increases in a dramatic way.
    The model is called morphable model because we can change some relationship among polygons in the model in order to create different appearances of the same face or also to create different faces. Shape and texture of the generic model are manipulated to adapt to the captured images.

Morphable models also allow to synthetize face expressions approximating the possible expressions of a specific subject.
The morphable model is based on a data set of 3D faces. Morphing between faces requires full correspondence (alignment) between all of the faces.

33. Face detection on **adaboost** vs face detection on **features**
    AdaBoost (Adaptive Boost) is a training technique that has the purpose of **learning the best sequence of weak classifiers** and their corresponding weights. A **linear classifier** is a classifier that tries to divide the two or more classes using only linear elements. A **Weak learner** is a classifier with a possibly very low accuracy (low as a coin toss). So we want to compose our strong classifier using weak learners. The **strong classifier** is represented by the combination or the sum of the values returned by the weak classifiers weighted according to its relevance in the final decision (they return a value in the set {-1, +1} according to the presence or not of a certain feature). In general -1 represents a negative outcome (non-face), while +1 is a positive detection (face). Once we carried out the weighted sum in order to limit ourself in the range (-1, +1), we've to divide this sum by the sum of weights.
34. How can I decide to accept or reject an identity claim?
    We can decide it according to the Acceptance Threshold.
35. **Genuine attempts** in the case of **open set identification**
    In the **open set identification** task the biometric system determines if the individual's biometric signature matches a biometric signature of someone in the gallery.
    Detection and Identification Rate is more or less equivalent to **Genuine Acceptance**; in fact, since the False Rejection is the complement of the GAR, we can compute it just doing: 1 - DIR(at rank 1) in the case of the Identification Open Set.
36. In multibiometric system, we want to fuse the result. We've two problems: which are they?
    Scores from different matchers are typically **non homogeneous**: Similarity/distance; Different ranges (eg. [0,1] o [0,100]); Different distributions.
    Also the **Reliability** of the different systems.
37. How many ways can we set the training and test set subsets?
    We can divide the DataBase into probe set, gallery set, train set and test set.
38. **Score level fusion**: three types of results to merge (class label, rank, score)
    The **Fusion Rules** that can be used in for the Score Level Fusion are Abstract, Rank and Measurement.
    The **Abstract Level** entails that the classifier just returns a class label to the input pattern. This class label represents he response of the classifier to a certain matching operation. One of the most way used to fuse the outcomes is to exploit **Majority Voting**: each classifier votes for a class, the pattern is assigned to the most voted class.
    The system may not return a single class label, but rather a **ranking of candidates**. In this case each classifier can produce a class ranking according to the probability of the pattern belonging to each of them. This means that higher the probability, the higher will be the position in the list. It's possible to use the **Borda Count**, where the ranking are then converted in scores that are summed up; the class with the highest final score is the one chosen by the multi-classifier.
    In the **Measurement**, each classifier outputs its classification score for the pattern in comparison with each class. The only problem we've to solve is to have values measurement that fall in the same range, so we must apply some kind of normalization. One trivial method is just to **sum** up the scores and to pass the result to the fusion rule. There is only one point that somehow occludes the full flexibility of

this approach that is the possibility of changing the kind of threshold that we exploit to get the final result.
Different methods are possible, including sum, weighted sum, mean, product, weighted product, max, min, ecc.

39. Which is the method for evaluating the distance between two minutiae?
    The **Ridge Count** is an abstract measure of the distance between any two any points of a fingerprint. Given two points a and b of a fingerprint, the ridge count is the number of ridge lines intersected by the segment ab. The number of ridges that separate two minutiae.
    Points a and b are often chosen as **relevant one** (e.g., core and delta). This cannot be done for each pair of minutiae because endpoints are not reliable enough (they can be caused by an interruption of one ridge due to bad thresholding or whatever).
40. **Problems** that arises in **iris recognition**
    The first problem is the **Reflection**. Reflection depends on the source of light and should be detected in advance. The presence of contact lens or glasses can influence the kind of reflection that can affect the iris. It is also important to consider when we have to do with a very dark iris, the texture of the iris may become completely invisible at the requested distance.
    Other possible problems are represented by: the **small size**; the **high resolution** required by the equipment; the **limited depth** of field so that we need to pay attention to the focus and out of focus problems; we need to **align with optical axis** unless we use some tricks like the so called "off-axis problem" (if the person is looking in another direction, the iris will not be exactly in the center of the sclera, but it will move towards the right or the left ends of the sclera); the **possible presence of glasses or contact lens**.
41. What can we use to solve the **dark iris problem**?
    When we have to do with a very dark iris, the texture of the iris may become completely invisible at the requested distance if we use the Visible Light capture modality.
    If we adopt near to **infrared sensors** we solve this problem because it is possible to highlight the texture of dark iris.
42. Which is the main difference between **PCA** and **LDA**
    Dimensionality reduction methods such as **Principal Component Analysis** (**PCA**) aim at reducing high-dimensional data sets to lower-dimensional data without significant information loss.
    The main difference between PCA and LDA is that the first one is **unsupervised**. When we collect samples to be used to build the new feature space, we don't know which is the subject that has been pictured in in each picture. We just collect a number of representative enough samples from different subjects and we try to create a new feature space over which we map each gallery image and each incoming probe before comparing them. The comparison happens in a reduced space so that we have a less sparse distribution of points and also both clustering and comparison are more efficient and more accurate.
    On the other hand, the **main limit of PCA** is that the variance over which the new projection relies can be either view to **inter-class differences** (and this is a good thing) but also to **intra-class differences**; so that, when we compute the eigenvectors, it can happen that some kind of variation belonging to the same subject in different situations is misclassified as the presence in a template of a different subject.

PCA is much more prone to intra-class misclassifications so it is more prone to false rejections in the sense that is is easier for PCA to be unable to recognize the same individual in a different situation.

The idea is to pass to **LDA that is a supervised method**; LDA is a supervised method because it takes into account not only the collections of samples but also a partitions of such samples into the different represented classes.

LDA is a technique of linear supervised (samples in the training set are labeled) dimensionality reduction whose objective is to maximize the separation between the classes.

In PCA we've a kind of unsupervised training, in the sense that we're not interesting in the real identity of the training subjects; all the training images are collected together in the same training set.

Now, in LDA, we rather **label the samples for each subject** that participates in the training set. This kind of partitioning helps to better learn which are the elements in the new space that better refer to intra-personal variation with the respect to those that actually refer to inter-personal variations that are those that we're interested in.

43. How do we evaluate the quality of a curved ROC?
    Using the **Area Under the Curve**.
44. **Cross validation** (subdivision train and test set)
    In **K Fold cross validation**, the data is divided into k subsets. The holdout method is repeated k times, such that each time, one of the k subsets is used as the test set/ validation set and the other k-1 subsets are put together to form a training set. The error estimation is averaged over all k trials to get total effectiveness of our model.
45. Why there are more filters in the **Gabor Filters**? How do they differ?
    In the **Feature Space** we can apply the **Gabor filters** (they are two dimensional filters that are applied on the image and what is maintained are the coefficients of convolution of images with those filters).
    The Gabor filters is a filter that is basically used for **edge detection**. It works as a **bandpass filter** (is a filter that captures only frequencies in a certain interval and cuts everything below or above the interval of frequency) for the local spatial frequency distribution, achieving an optimal resolution in both spatial and frequency domains.
    A feature vector is obtained by convolution of an image with a **Gabor filter bank** (different orientations and different scales). Each of the image in this bank is a filter that is convolved with the original image in order to capture different kinds of pattern.
    In the first row, what is change is the orientation of the filter; on the other hand, we can change the scale of the filter that represents the spatial frequency. Combining different orientations and scale, it’s possible to obtain a huge bank of filters.
    We compute the convolution of these filters with different regions of the image; we slide these convolution filters over the image on we compute the response in order to detect the presence of patterns of this kind.
    2D Gabor functions **enhance edge contours**. This corresponds to enhancing eye, mouth, and nose edges, and also moles, dimples, scars, etc. If the convolution is performed on all the pixel of the image and we take the overall result, dimensionality is hight (size of the image). In alternative it can be performed on a regular grid of points, or on salient face regions only.
    Going from top to bottom, we obtain details of different size. This kind of process is **computationally expensive** because we’ve to carry out the convolution as many times as is the size of the cardinality of the filter bank.
    Moreover, there is another problem: the result of each convolution is itself an image of

the same size of the original one, so that this can be considered as a **feature based representation of the original image**.

46. What is the variant of LBP that allows you to optimize the length of the histograms?
    The **Uniform pattern**. They are relevant since they identify significant structures.
    We consider as uniform all patterns where there are a maximum of two changes from 0 to 1.